Universidade Federal de Campina Grande
Centro de Engenharia Elétrica e Informática
Programa de Pós-Graduação em Ciência da Computação

Estimativa de Esforço em *Story Point* a partir do Texto da *User Story* com Aprendizagem de Máquina e LLM

Giseldo da Silva Néo

Tese submetida à Coordenação do Curso de Pós-Graduação em Ciência da Computação da Universidade Federal de Campina Grande (Campus I) como parte dos requisitos necessários para obtenção do grau de Doutor em Ciência da Computação.

Área de Concentração: Ciência da Computação
Linha de Pesquisa: Metodologia e Técnicas da Computação

Dr. JOSÉ ANTÃO BELTRÃO MOURA
(Orientador)

Campina Grande, Paraíba, Brasil

MINISTÉRIO DA EDUCAÇÃO
UNIVERSIDADE FEDERAL DE CAMPINA GRANDE
POS-GRADUACAO EM CIENCIA DA COMPUTACAO
Rua Aprígio Veloso, 882, Edifício Telmo Silva de Araújo, Bloco CG1, - Bairro Universitário, Campina Grande/PB, CEP 58429-900
Telefone: 2101-1122 - (83) 2101-1123 - (83) 2101-1124
Site: http://computacao.ufcg.edu.br - E-mail: secpg@computacao.ufcg.edu.br

FOLHA DE ASSINATURA PARA TESES E DISSERTAÇÕES

GISELDO DA SILVA NEO

ESTIMATIVA DE ESFORÇO EM STORY POINT A PARTIR DO TEXTO DA USER STORY COM APRENDIZAGEM DE MÁQUINA E LLM

Tese apresentada ao Programa de Pós-Graduação em Ciência da Computação como pré-requisito para obtenção do título de Doutor em Ciência da Computação.

Aprovada em: 16/09/2025

Prof. Dr. JOSÉ ANTÃO BELTRÃO MOURA, Orientador, UFCG

Prof. Dr. FÁBIO JORGE ALMEIDA MORAIS, Examinador Interno, UFCG

Prof. Dr. EVANDRO DE BARROS COSTA, Examinador Interno, UFAL

Prof. Dr. THALES MIRANDA DE ALMEIDA VIEIRA, Examinador Externo, UFAL

Prof. Dr. TÁRCIO RODRIGUES BEZERRA, Examinador Externo, IFAL

Documento assinado eletronicamente por **JOSE ANTAO BELTRAO MOURA**, **PROFESSOR 3 GRAU**, em 04/05/2026, às 17:44, conforme horário oficial de Brasília, com fundamento no art. 8º, caput, da Portaria SEI nº 002, de 25 de outubro de 2018.

Documento assinado eletronicamente por **THALES MIRANDA DE ALMEIDA VIEIRA**, **Usuário Externo**, em 05/05/2026, às 14:42, conforme horário oficial de Brasília, com fundamento no art. 8º, caput, da Portaria SEI nº 002, de 25 de outubro de 2018.

Documento assinado eletronicamente por **FABIO JORGE ALMEIDA MORAIS**, **PROFESSOR(A) DO MAGISTERIO SUPERIOR**, em 05/05/2026, às 15:21, conforme horário oficial de Brasília, com fundamento no art. 8º, caput, da Portaria SEI nº 002, de 25 de outubro de 2018.

Documento assinado eletronicamente por **Tárcio Rodrigues Bezerra**, **Usuário Externo**, em 07/05/2026, às 20:58, conforme horário oficial de Brasília, com fundamento no art. 8º, caput, da Portaria SEI nº 002, de 25 de outubro de 2018.

A autenticidade deste documento pode ser conferida no site https://sei.ufcg.edu.br/autenticidade, informando o código verificador **6340177** e o código CRC **52C479BF**.

Universidade Federal de Campina Grande - UFCG
Sistema de Bibliotecas - SISTEMOTECA
Catalogação de Publicação na Fonte. UFCG - Biblioteca Central

N438e Néo, Giseldo da Silva.

Estimativa de esforço em *Story Point* a partir do texto da *User Story* com aprendizagem de máquina e LLM / Giseldo da Silva Néo. – 2026.

306 f. : il. color.

Tese (doutorado em Ciência da Computação) – Universidade Federal de Campina Grande, Centro de Engenharia Elétrica e Informática, 2025.

"Orientação: Prof. Dr. José Antão Beltrão Moura".

Referências.

1. Estimativa de Esforço. 2. Large Language Models. 3. *Story point*. 4. *User story*. I. Moura, José Antão Beltrão. II. Título.

UFCG/BC CDU 004(043.2)

FICHA CATALOGRÁFICA ELABORADA PELA BIBLIOTECÁRIA ITAPUANA SOARES DIAS GONÇALVES CRB-15/93

Agradecimentos

Agradeço especialmente ao orientador, Dr. J. Antão B. Moura, por seus inestimáveis conselhos e orientações. A todos os professores que participaram da banca de qualificação e defesa, além dos colegas estudantes.

Agradeço à minha amada esposa, Alana, e às minhas maravilhosas filhas, Gisella e Alice. Aos pais dedicados, Irene e Giselio, além dos irmãos que me inspiram, Alex e Michelle. Também agradeço aos amigos e familiares que me apoiaram nesta jornada. Por fim, agradeço aos servidores (Professores e técnicos) da Universidade Federal de Campina Grande e da Universidade Federal de Alagoas, e a todos que contribuíram para este doutoramento.

Agradeço com entusiasmo ao Instituto Federal de Alagoas (IFAL) e a seus servidores que permitiram e incentivaram a realização desta tese, representados pela Pró-Reitora de Pesquisa, Pós-Graduação e Inovação do IFAL, a Profa. Eunice Palmeira da Silva.

Estendo meus agradecimentos a todos que, de forma direta ou indireta, colaboraram com ideias, discussões e sugestões que enriqueceram o conteúdo desta Tese.

Resumo

A estimativa de esforço em projetos ágeis de software continua sendo um desafio persistente na indústria, especialmente quando se utilizam artefatos textuais como *User Stories* para prever os Story Points. Esta tese investiga o uso de técnicas de Processamento de Linguagem Natural (PLN) e Aprendizagem de Máquina (AM) na previsão de esforço, considerando as descrições textuais das *User Stories* como a principal fonte de informação. Inicialmente, uma revisão sistemática da literatura identificou técnicas predominantes, como Term Frequency – Inverse Document Frequency (TF-IDF) combinado com Support Vector Machine (SVM), e destacou lacunas relacionadas ao uso de atributos de legibilidade, sentimento e subjetividade, bem como à aplicação de Large Language Models (LLMs) nessa tarefa. A pesquisa propôs três abordagens principais: (i) o *Neo Legibility Effort Model*, que utiliza atributos extraídos automaticamente do texto da *User Story* para prever o esforço; (ii) o Neo *User Story* Tutor, uma aplicação baseada em LLMs para sugerir melhorias na escrita das *User Stories*, visando a maior precisão nas estimativas; e (iii) o Neo LLM Predictor, que utiliza LLMs para estimar diretamente os *Story Points* com diferentes estratégias (few-shot, zero-shot e fine-tuning). Para suportar os experimentos, foi construído o NeoDataset, um novo conjunto de dados coletado a partir de projetos reais hospedados no GitLab. Os modelos propostos foram avaliados com métricas, como o Mean Absolute Error (MAE), e comparados com *baselines* consagrados na literatura. Os resultados demonstraram que tanto os atributos de legibilidade quanto os LLMs podem contribuir significativamente para a melhoria das estimativas de esforço em ambientes ágeis. A tese apresenta evidências de que é possível aumentar a acurácia das estimativas por meio da combinação de análise textual e aprendizado de máquina, além de destacar a relevância de aspectos linguísticos na qualidade das *User Stories*.

Abstract

Effort estimation in agile software projects remains a persistent challenge in the industry, especially when using textual artifacts such as *User Stories* to predict Story Points. This thesis investigates the use of Natural Language Processing (NLP) and Machine Learning (ML) techniques in effort prediction, considering the textual description of *User Stories* as the main source of information. Initially, a systematic literature review identified prevalent techniques for the said estimation, such as Term Frequency – Inverse Document Frequency (TF-IDF) combined with Support Vector Machine (SVM), and highlighted gaps related to the use of readability, sentiment, and subjectivity attributes, as well as the sacant of application of Large-Scale Language Models (LLMs) for this task. The research proposed and evaluated three main approaches: (i) the *Neo Legibility Effort Model*, which uses attributes automatically extracted from *User Story* text to predict effort; (ii) the Neo *User Story* Tutor, an LLM-based application that suggests improvements in *User Story* writing to improve estimation accuracy; and (iii) the Neo LLM Predictor, which uses LLMs to directly estimate *Story Points* using different strategies (few-shot, zero-shot, and fine-tuning). To support the experiments a new dataset collected from real projects hosted on GitLab, was built (aka NeoDataset). The proposed models were evaluated using metrics such as MAE and compared with established *baselines* in the literature. The results demonstrated that both readability attributes and LLMs can significantly contribute to improving effort estimates in agile environments. The thesis presents evidence that it is possible to increase estimate accuracy through the combination of textual analysis and machine learning, in addition to highlighting the relevance of linguistic aspects in the quality of *User Stories*.

Lista de Figuras

Lista de Tabelas

Lista de Abreviaturas e Siglas

Lista de Abreviaturas e Siglas

AM	Aprendizagem de Máquina
API	Application Programming Interface
ASD	Agile Software Development
BERT	Bidirectional Encoder Representations from Transformers
BoW	Bag of Words
CBIE	Congresso Brasileiro de Informática na Educação
CLI	Coleman–Liau Index
CNN	Convolutional Neural Network
CSV	Comma-Separated Values
DL	Deep Learning
EEDS	Estimativa de Esforço de Desenvolvimento de Software
ELMo	Embeddings from Language Models
EM	Engenharia de Machine Learning
FRE	Flesch Reading Ease
FN	False Negative
FP	False Positive
GPT	Generative Pre-trained Transformer
GPU	Graphics Processing Unit
IA	Inteligência Artificial
JSON	JavaScript Object Notation
KNN	K-Nearest Neighbors

LDA	Latent Dirichlet Allocation
LLM	Large Language Model
LM	Language Model
LR	Logistic Regression
LSTM	Long Short-Term Memory
MAE	Mean Absolute Error (Erro Absoluto Médio)
MbR	Mean-based Regression
ML	Machine Learning
MLP	Multi-Layer Perceptron
MMRE	Mean Magnitude of Relative Error
MSE	Mean Squared Error
NER	Named Entity Recognition
NLTK	Natural Language Toolkit
NLP	Natural Language Processing
PLN	Processamento de Linguagem Natural
PMBOK	Project Management Body of Knowledge
PPL	Perplexity
PF	Ponto de Função
QA	Question Answering
QG	Question Generation
RAG	Retrieval-Augmented Generation
ReLU	Rectified Linear Unit
RL	Regressão Linear
RMSE	Root Mean Square Error
RNN	Recurrent Neural Network
RoBERTa	Robustly Optimized BERT Pretraining Approach
RQ	Research Question
SCRUM	Framework de Desenvolvimento Ágil
SDEE	Software Development Effort Estimation
SP	Story Points
SVM	Support Vector Machine

SVR	Support Vector Regression
TF-IDF	Term Frequency–Inverse Document Frequency
TAM	Technology Acceptance Model
TN	True Negative
TP	True Positive
UDA	Unsupervised Domain Adaptation
UP	Use Case Points
UST	User Story Tutor
XAI	Explainable Artificial Intelligence
XP	Extreme Programming

Conteúdo

Capítulo 1

Introdução

O valor de uma formação universitária não reside no aprendizado de muitos fatos, mas no treinamento da mente para conceber coisas novas.

Albert Einstein

1.1 Contexto e motivação

Estimar a duração de projetos de software é um processo desafiador. O *Standish Group International* publica a cada dois anos um relatório que confirma essa dificuldade [139]. Esse relatório, chamado de *CHAOS Report*, consolida a taxa de sucesso de mais de 50 mil projetos de tamanhos, empresas e segmentos diferentes e apresenta esses dados na forma de quadros e tabelas. Outro relatório da mesma empresa, mais focado em uma análise textual subjetiva, é o *CHAOS Manifesto*.

Nesse último relatório, metade dos projetos foram classificados em relação ao atributo SITUAÇÃO, como desafiadores [112, 138]. A SITUAÇÃO, no *CHAOS Report*, é um atributo do tipo qualitativo policotômico e pode assumir um dos três valores: *SUCESSO*, *DESAFIADOR* ou *FALHOU*. Esse campo SITUAÇÃO é um *proxy* derivado de três outros indicadores: *OnTime, OnBudget e OnTarget*. Por exemplo: se o projeto foi resolvido em um prazo estimado razoável (*OnTime*), razoavelmente dentro do orçamento (*OnBudget*) e com grande parte do escopo planejado (*OnTarget*), seu atributo situação será classificado como *SUCESSO*.

Na tabela 1.1 são apresentados os dados do *CHAOS Report* de 2011 a 2015 e do ano de 2021 [1]. Analisando esses dados, nota-se que a maioria dos projetos teve uma SITUAÇÃO diferente de SUCESSO; portanto, não alcançou o término no prazo, custo e escopo definidos em seu planejamento. No entanto, é necessário ter cuidado ao generalizar a partir do resumo dos dados deste relatório, pois cada projeto é único por definição [109]; além disso, o conceito de *SUCESSO* usado no relatório pode ter interpretações diferentes entre as partes interessadas.

Tabela 1.1: Taxa de sucesso dos projetos.

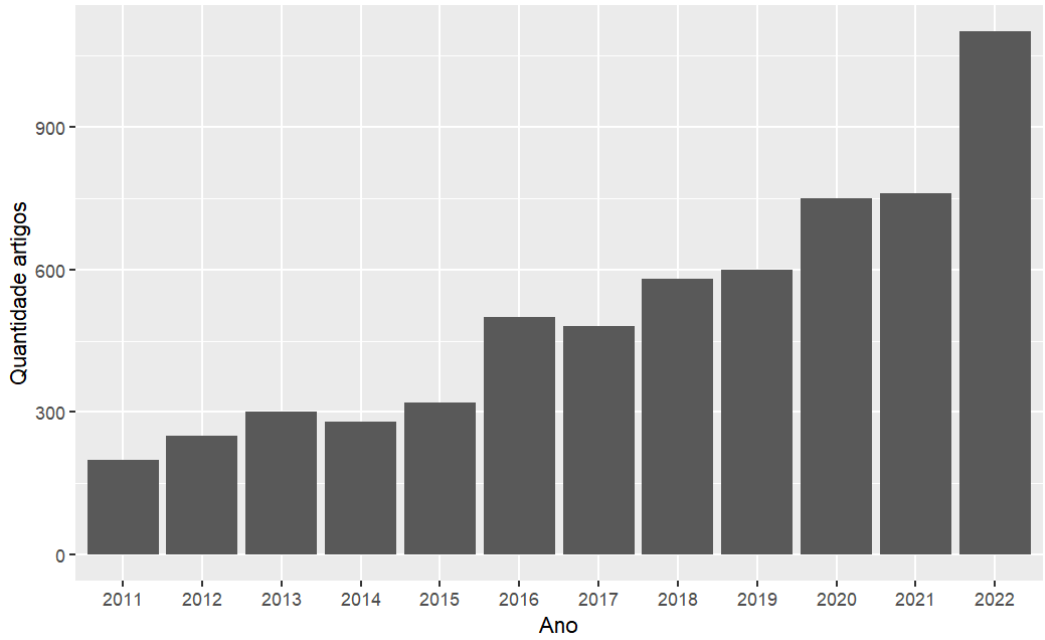
SITUAÇÃO	**2011**	**2012**	**2013**	**2014**	**2015**	**2021**
SUCESSO	39%	37%	41%	36%	36%	31%
DESAFIADOR	39%	46%	40%	47%	45%	50%
FALHOU	22%	17%	19%	17%	19%	19%

Concomitantemente à taxa de relativo fracasso dos projetos, pesquisas relacionadas a estimativas de software aumentaram consideravelmente desde 2011 (ano do primeiro *CHAOS Report*), conforme os dados fornecidos pela empresa *Wizdom.AI* (figura 1.1) [171]. Essa empresa monitora continuamente bilhões de dados sobre o ecossistema de pesquisas científicas. Os dados são oriundos de mais de 110 milhões de publicações dos principais centros de pesquisa internacionais [171]. Verifica-se (figura 1.1) uma tendência de crescimento de artigos que contêm o termo *software development effort estimation* nas palavras-chave ao longo do tempo [171]; isso sinaliza que o interesse da indústria e de pesquisadores no tema vem crescendo continuamente.

Apesar do interesse do mercado e dos pesquisadores no tema da estimativa de esforço de software, alguns gerentes acreditam que estimar o esforço de software é um problema complexo e sem solução [161]. Porém, mesmo com a descrença de alguns profissionais e, por outro lado, com evidências empíricas de sua taxa de acerto, as estimativas ainda são utilizadas estrategicamente nas organizações. Um exemplo do seu uso é na definição de valores contratuais do governo brasileiro, que utiliza uma estimativa de tamanho funcional, o Ponto de Função, como referência

[1] Estes são os anos do relatório disponibilizados gratuitamente na internet. Para ter acesso aos outros anos, é necessário pagar uma taxa ou ser afiliado a organização *Standish Group International*

Figura 1.1: Artigos com *software development effort estimation* em suas palavras chave.



para medir e remunerar contratos públicos de desenvolvimento e manutenção de sistemas [86]. Essa medida funcional utiliza os requisitos existentes em várias fases do projeto como base para o cálculo do esforço, sendo classificada como uma técnica do tipo algorítmica para estimar o tamanho do software [141].

Outra medida de tamanho funcional, além do Ponto de Função, é o *Story Point* , utilizado em projetos que adotam práticas ágeis (*Agile*). Essas práticas ágeis vêm sendo utilizadas cada vez mais por empresas que desenvolvem projetos de software. Elas podem fornecer flexibilidade na entrega de produtos por meio de várias iterações com valor agregado [26]. Para um projeto de software ser aderente às práticas ágeis, a equipe deve seguir os princípios encontrados no manifesto ágil [2]. Seu foco é em entregas de valor e estimativas que utilizam uma documentação mínima, porém suficiente.

Paralelamente, Modelos de linguagem de larga escala (em inglês, Large Language Models - LLMs) surgiram como ferramentas promissoras para lidar com tarefas complexas de processamento de linguagem natural [115]. Modelos pré-treinados, como BERT e GPT, demonstraram capacidade de compreender nuances semânticas em texto e transferir esse conhecimento para diversas tarefas específi-

cas através de *Fine-Tuning* [85].

1.2 Problema

Nos times que utilizam práticas ágeis, uma estimativa de tamanho funcional bastante utilizada é o *Story Point* (apesar de existirem outras, como dias ideais [26]). Ele é calculado a partir da análise da *User Story* [164]. *User Stories* são textos em linguagem natural que definem o escopo do que será construído e são elaborados pela equipe junto com o patrocinador, além de serem registrados em cartões ou em algum software apropriado [124]. A estimativa precisa do *Story Point* é um elemento-chave para o sucesso do projeto no desenvolvimento ágil de software; no entanto, a estimativa manual apresenta limitações, como subjetividade e inconsistência [88]. Para o cálculo do *Story Point* de uma *User Story*, algumas técnicas podem ser utilizadas, entre elas, planning poker, opinião de especialistas e, mais recentemente, a aprendizagem de máquina (AM) [141].

A AM é uma subárea de inteligência artificial (IA). Em uma definição mais formal, é a capacidade de aumentar o desempenho na realização de alguma tarefa por meio da experiência [87]. Além disso, a AM está relacionada à construção de modelos preditivos e a estudos na área de processamento de linguagem natural.

O uso da AM na estimativa de esforço é uma técnica que tem ganhado destaque tanto em estudos quanto nas empresas [32]. Em AM, um modelo preditivo é um software que pode prever valores, sendo treinado com dados históricos. No domínio da engenharia de software e da estimativa de esforço, vários pesquisadores já propuseram diversos modelos preditivos [23, 130, 111, 1], definindo qual algoritmo preditivo ou técnica de pré-processamento com desempenho superior deve ser utilizado em diferentes situações de estimativa de esforço (por exemplo, características do software, equipe, etc.). Porém, seus resultados ainda não são conclusivos [150], o que sugere um problema de *instabilidade na conclusão* [83], que é quando vários pesquisadores chegam a conclusões diferentes para um mesmo problema, em parte por causa de mudanças nos dados utilizados ou na própria metodologia utilizada.

Uma dessas abordagens de modelos preditivos, que estabeleceu um *baseline* para estimativa de esforço, é chamada de TFIDF-SE [2] de Porru et al. [111]. Eles utilizaram *term frequency–inverse document frequency* (TF-IDF) para representar o texto da *User Story* e para o algoritmo preditor a Máquina de Vetores de Suporte (em inglês, *Support Vector Machine*, sigla SVM) como algoritmo preditivo.

Quando se utiliza texto como atributo preditor, como no caso da estimativa de esforço a partir do texto da *User Story*, é preciso representar esse texto quantitativamente de alguma forma. No caso do preditor TFIDF-SE, de Porru et al. [111], foi utilizada a técnica TF-IDF; porém, existem outras técnicas que carregam mais contexto nessa representação.

Essas técnicas de representação de texto variam desde métodos mais simples, como *Bag-of-Words* (BoW) e *TF-IDF*, que contam a frequência das palavras sem considerar seu contexto, até abordagens mais sofisticadas, como *N-grams*, que capturam sequências de palavras. As técnicas mais avançadas, como *Word Embeddings* (ex: Word2Vec) e *Embeddings Contextuais* (ex: BERT), representam palavras como vetores em espaços de baixa dimensão, onde a proximidade dos vetores indica similaridade semântica, permitindo que os modelos compreendam o significado e o contexto das palavras de forma muito mais rica. A escolha da técnica mais adequada depende da complexidade da tarefa e dos recursos disponíveis. Veja na Figura 1.2 uma evolução sequencial das técnicas.

Figura 1.2: Algumas técnicas de representação de texto

Outro modelo de estimativa de esforço proposto foi o DEEP-SE, de Choetkiertikul et al. [23], que utilizou redes neurais recorrentes (RNNs), especificamente *long short-term memory* (LSTM) e *recurrent highway network* e comparou sua abordagem com o TFIDF-SE de Porru et al. [111]. No estudo em questão, concluiu-se que o DEEP-SE obteve aparente sucesso em relação à abordagem anterior escolhida como *baseline* (o TFIDF-SE).

[2] SE, no TFIDF-SE é a sigla para *Software Effort*

Porém, algum tempo depois, Tawosi et al. [150] reproduziram os experimentos de Porru et al. [111] e Choetkiertikul et al. [23] e chegaram à conclusão de que o modelo DEEP-SE não supera o preditor TFIDF-SE. Ou seja, o TF-IDF-SE de Porru et al. [111] ainda é um *baseline* a ser superado.

1.3 Lacuna

Conforme mencionado, existem algumas técnicas para representação de texto para uso nos preditores de estimativa de esforço que utilizam o texto da *User Story* como atributo preditor do esforço. Algumas dessas técnicas são o TF-IDF e Embeddings Contextuais [23, 111].

Porém, existem técnicas de extração de atributos quantitativos do texto que são mais rápidas e menos onerosas, que também podem ser utilizadas. Alguns desses atributos quantitativos que podem ser extraídos do texto são os indicadores de legibilidade do texto [53], além do sentimento e da subjetividade. Estes atributos podem ser extraídos e utilizados na predição.

Esses atributos de legibilidade (em inglês, *readability*) já foram utilizados para prever a capacidade de entrega de equipes que desenvolvem projetos de software em outro estudo do mesmo criador do DEEP-SE [22], especificamente o atributo *Complexity of description*, que foi extraído do índice de legibilidade *Gunning Fog*. Além disso, a legibilidade vem sendo utilizada em outros domínios com resultados promissores, tais como seu uso na contabilidade para prever características de demonstrativos contábeis [44].

A utilização de modelos *de Linguagem em Grande Escala (Large Language Models* - LLMs) para a estimativa de esforço no desenvolvimento de software também apresenta uma lacuna a ser explorada. Investigar o uso de LLMs para essa tarefa preditiva pode revelar abordagens mais robustas, automáticas e contextualmente sensíveis, ampliando as possibilidades de apoio à tomada de decisão em processos ágeis de software, especificamente na estimativa de esforço.

Portanto, dadas as questões de pesquisa da seção anterior e este contexto, as lacunas que oferecem oportunidades de pesquisa para abordar as questões de

pesquisa de interesse identificadas nesta tese são:

Lacuna 1: A literatura revisada indica que, ainda não foi validado o uso dos atributos: legibilidade, sentimento e subjetividade para prever a estimativa de esforço em *Story Points* a partir das *User Stories*.

Lacuna 2: Aparentemente, *Large Language Models* podem ser utilizados como preditores de estimativa de esforço em *Story Points* e para sugerir recomendações no texto da User Story.

1.4 Hipótese

Diante do exposto, apresentam-se as hipóteses:

Hipótese 1: É possível estimar (com precisão comparável ou maior do que os baselines - citados anteriormente) os *Story Points* a partir do texto das *User Stories*, extraindo os atributos: legibilidade, subjetividade e sentimento do texto da User Story.

Hipótese 2: É possível melhorar o texto das *User Stories* com recomendações oriundas de Modelos de Linguagem de Grande Porte (LLM).

Hipótese 3: É possível prever a estimativa em *Story Points* com *Large Language Models* (com precisão comparável ou maior que os baselines).

1.5 Problemas e Objetivos da Pesquisa

1.5.1 Problema de Negócio

A estimativa de esforço em projetos de software é reconhecida como um desafio da engenharia de software contemporânea. Relatórios como o CHAOS Report evidenciam que mais da metade dos projetos de software são entregues fora do prazo ou com custos superiores aos planejados, o que impacta diretamente a sustentabilidade financeira e operacional das organizações. Mesmo com a adoção de metodologias ágeis, que buscam maior adaptabilidade e entrega contínua de valor, a estimativa manual de *Story Points* permanece fortemente dependente do

julgamento humano, conduzindo a resultados inconsistentes e subjetivos. Essa imprecisão afeta o planejamento das sprints, a precificação de contratos e a alocação de recursos, ocasionando prejuízos e retrabalho nas equipes de desenvolvimento.

1.5.2 Problema Técnico e Científico

Do ponto de vista técnico, o problema central reside na dificuldade de construir modelos preditivos confiáveis e generalizáveis para estimar o esforço (em Story Points) a partir do texto das *User Stories*. De fato, este é um problema técnico antigo e continuado tendo sido apontado, inclusive, como o mais crítico, em survey com praticantes Scrum, conforme resultados de Andrade [7]. Embora existam modelos tradicionais baseados em TF-IDF e Support Vector Machine (como o TFIDF-SE de Porru et al. [111]) e abordagens com redes neurais recorrentes (como o DEEP-SE de Choetkiertikul et al. [23]), seus resultados são inconsistentes [150] e dependem fortemente do conjunto de dados e do contexto do projeto. Persistem ainda lacunas técnicas relacionadas a:

- O uso limitado de atributos linguísticos na representação quantitativa das *User Stories*;
- A escassa exploração de LLMs como preditores de esforço ou como agentes de melhoria textual;
- A instabilidade dos resultados entre estudos que reproduzem os mesmos experimentos indica a necessidade de novos conjuntos de dados e de modelos mais robustos e interpretáveis.

Diante desse cenário, o desafio científico consiste em investigar e validar métodos - dentre eles os de aprendizagem de máquina e LLMs, capazes de reduzir o erro de previsão de Story Points, explorando atributos linguísticos e semânticos extraídos automaticamente do texto da User Story.

1.5.3 Objetivo Geral

Investigar, desenvolver e avaliar modelos preditivos baseados em Aprendizagem de Máquina e Modelos de Linguagem de Grande Escala (LLMs) para estimar *Story*

Points a partir do texto em linguagem natural das *User Stories*, de modo a reduzir a subjetividade e aumentar a precisão das estimativas de esforço em projetos ágeis.

1.5.4 Objetivos Específicos

- Propor um modelo preditivo que utilize atributos linguísticos — legibilidade, sentimento e subjetividade — extraídos automaticamente do texto da *User Story*, comparando seu desempenho com *baselines* reconhecidos na literatura (TFIDF-SE e média dos Story Points).
- Desenvolver uma aplicação de suporte, baseada em LLMs, para recomendar melhorias textuais nas *User Stories*, visando aumentar a clareza e reduzir a ambiguidade das descrições.
- Avaliar o uso de LLMs como preditores diretos de *Story Points*, utilizando estratégias few-shot, zero-shot e *Fine-Tuning* e comparando os resultados com modelos tradicionais de aprendizado de máquina.
- Construir e disponibilizar um novo conjunto de dados extraído de projetos reais do GitLab, garantindo diversidade e representatividade para os experimentos.
- Comparar os modelos propostos quanto a métricas de desempenho (MAE), interpretabilidade e aplicabilidade prática no contexto de equipes ágeis.
- Minimizar o problema de instabilidade dos resultados, utilizando múltiplos conjunto de dados e replicações experimentais, fortalecendo a validade externa dos achados.

Veja na Tabela 1.2 um resumo da relação entre problema de negócio e técnico e os objetivos da pesquisa.

1.6 Questões de pesquisa

As questões de pesquisa que orientam esta Tese são:

- QP 1: Qual é o estado da arte para prever *Story Points* a partir do texto das *User Stories*? Esta QP1 será respondida com a revisão sistemática (Apêndice C).

Tabela 1.2: Relação entre problemas e objetivos da pesquisa.

Dimensão	Descrição	Correspondência
Problema de Negócio	Estimativas imprecisas em projetos ágeis prejudicam custo e prazo.	Justifica a relevância prática da pesquisa
Problema Técnico e Científico	Modelos atuais não exploram atributos linguísticos nem LLMs de forma robusta.	Explora a lacuna científica e tecnológica.
Objetivo Geral	Investigar e propor modelos de ML e LLM para prever *Story Points* com base no texto.	Direciona o propósito central da tese.
Objetivos Específicos	Implementação, avaliação e validação experimental.	Estrutura as etapas metodológicas.

- QP 2: Como melhorar o texto das *User Stories* para que possam ser usadas em estimativas de *Story Points* para redução de erro? Respondida com o uso de LLMs (Capítulo 6).
- QP 3: Qual técnica pode ser utilizada para uma estimativa que supere os *baselines* identificados (menor erro) para prever *Story Points* a partir do texto das *User Stories*? Respondida com o uso de legibilidade (Capítulo 7) e ajuste-fino de LLM (Capítulo 8).

Para endereçar a questão QP 1, foram mapeadas as principais técnicas utilizadas por outros pesquisadores no tema, identificando os *baselines* para comparação e avaliando as principais soluções. Nas questões seguintes (QP 2 e QP 3), esta pesquisa de doutorado avaliou a recomendação de melhoria do texto da *User Story*, a efetividade do modelo preditor sugerido baseado na extração dos atributos e no preditor com LLM. Para organizar o tratamento das QP 1, QP 2 e QP 3, cada uma delas foi decomposta em subquestões, como segue.

- QP 1: Qual é o estado da arte para prever *Story Points* a partir do texto das *User Stories*?
  - QP 1.1: Quais são as técnicas de PLN e AM utilizadas na estimativa de esforço de desenvolvimento de software?
    - A motivação para criar e responder a esta QP 1.1 foi identificar quais são as técnicas aplicadas nos modelos preditivos de estimativa de

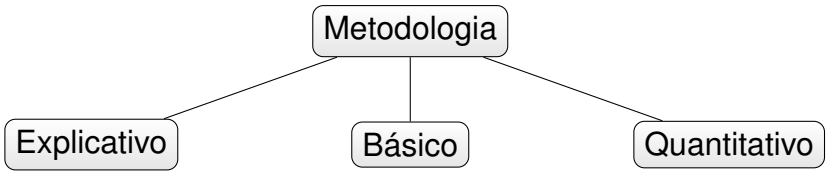
esforço, a fim de encontrar lacunas e oportunidades de avanço.

  - QP 1.2: Quais são os artefatos textuais utilizados na estimativa?
    - A motivação foi entender quais são os artefatos textuais mais utilizados para formular uma abordagem, utilizando como entrada os artefatos textuais mais comuns.
  - QP 1.3: Quais são as métricas utilizadas para avaliação de avaliação utilizadas?
    - A motivação foi identificar as métricas mais adequadas e comparáveis utilizadas nesse domínio para utilizá-las no método de construção do modelo preditivo.

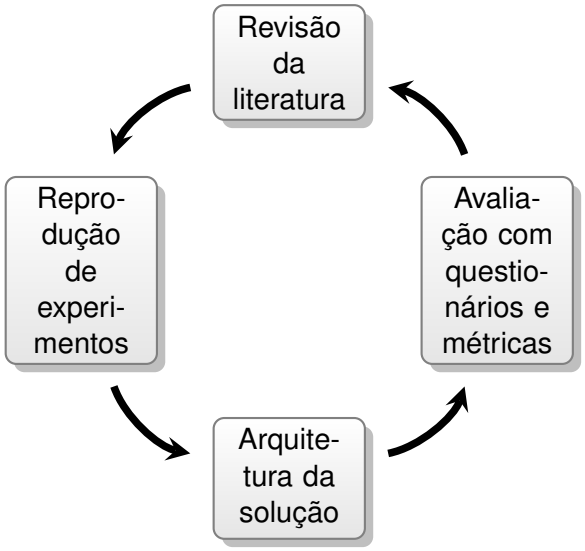
- QP 2: Como melhorar o texto das *User Stories* para que possam ser utilizadas em estimativas de *Story Points* para redução de erro?
  - QP 2.1: É possível melhorar a estimativa recomendando mudanças no texto da User Story?
    - A motivação foi melhorar a estimativa em Story Point, recomendando uma melhoria de texto da *User Story* com o ChatGPT.
- QP 3: Qual técnica pode ser utilizada para uma estimativa que supere os *baselines* identificados para prever *Story Points* a partir do texto das *User Stories*?
  - QP 3.1: Quão bem o modelo com extração de atributos performa em comparação com os baselines?
    - A motivação foi avaliar a efetividade do modelo comparando-o com uma técnica simples, a média dos valores dos Story Points.
  - QP 3.3: O modelo com LLM tem menor erro em comparação com os baselines?
    - A motivação foi avaliar a efetividade do modelo comparando-o com os baselines. A motivação foi avaliar se há melhoria com o uso de LLM na estimativa.

1.7 Metodologia

O objetivo da Tese é explicativo e experimental [120], pois deseja-se entender a relação de causa e efeito entre o texto da *User Story* e seu respectivo número de Story Points. Quanto à natureza, trata-se de uma pesquisa aplicada, e quanto à análise dos resultados, é quantitativa (figura 1.7).



Em relação aos procedimentos, de forma geral, foram realizados os seguintes passos: uma revisão da literatura; reprodução de experimentos de outros pesquisadores; construção dos experimentos a partir das hipóteses; por fim, avaliação com as métricas mais adequadas (figura 1.7).



Em detalhes, os procedimentos realizados foram:

- Uma revisão da literatura em periódicos relacionados ao tema;
- Reproduções dos experimentos de outros pesquisadores: Scott and Pfahl [130], Choetkiertikul et al. [23], Porru et al. [111], Tawosi et al. [150];
- Coletas de dados de projetos de software com seus respectivos *User Stories* e *Story Points* oriundos do repositório on-line do GitLab [48]. Para isso, foi construída uma aplicação em Python. O conjunto de dados assim coletados

chama-se *NeoDataset*;

- Implementação de um modelo preditivo (chamado *Neo Legibility Effort Model*) com Python e com o suporte da biblioteca *scikit-learn*, utilizando o fluxo de aprendizagem de máquina do Google [50].
- Avaliação comparativa do *Neo Legibility Effort Model* contra dois baselines, a média dos *Story Points* e o TFIDF-SE. Para a métrica de comparação, foi utilizado o Erro Médio Absoluto;
- Construção de uma aplicação web chamada *Neo User Story Tutor* , com um módulo de recomendação das *User Stories* que utiliza os Large Language Models.
- Avaliação do *Neo User Story Tutor* com um questionário apoiado em dois métodos de avaliação: *AttrakDiff* e o *Technology Acceptancce Model*.
- Construção de um modelo preditivo de previsão de *Story Points Neo LLM predictor* que utiliza o Large Language Models.
- Avaliação do *Neo LLM predictor* utilizando as técnicas few-shot e zero-shot.

1.8 Contribuições

As principais contribuições da tese incluem o desenvolvimento e a avaliação de três experimentos para a estimativa de esforço e a melhoria da qualidade das *User Stories* em contextos ágeis. Um deles, o Neo *User Story* Tutor, mostrou potencial para recomendar ajustes no texto das *User Stories* com base em métricas linguísticas, auxiliando os desenvolvedores na escrita mais clara e objetiva. O *Neo Legibility Effort Model* demonstrou que atributos de legibilidade e complexidade textual podem ser utilizados para prever o esforço com consistência. Já o Neo LLM Predictor, com modelos LLM via fine-tuning, alcançou desempenho competitivo e erros reduzidos na previsão de Story Points, contribuindo como ferramenta complementar a métodos tradicionais, como o planning poker.

Foram publicados 5 estudos derivados da tese em conferências e periódicos, os quais estão disponíveis no Apêndice [95], [94], [96], [97] e [29]. Além disso, resultados parciais dos estudos publicados já foram utilizados em pesquisas por

outros pesquisadores, conforme a lista a seguir.

- Zou et al. [175] utilizaram o NeoDataset como fonte de dados brutos a partir da qual um novo conjunto de dados foi construído, por meio de um método de construção semiautomático. O NeoDataset foi escolhido devido à sua oportunidade (timeliness) e ao fato de que seus dados são extraídos diretamente dos painéis de problemas (issue boards) do GitLab, garantindo forte consistência e estrito alinhamento com cenários do mundo real. Os dados brutos extraídos foram pré-processados, removendo registros com informações insuficientes e *User Stories* com 10 ou menos palavras de contexto. Depois, alguns atributos foram extraídos do texto, tais como: Metas (goals); Atores (actors); Impactos (impacts); Entregáveis (deliverables); *User Stories*; e informações correspondentes ao contexto do projeto.
- Moon et al. [88] empregaram o NeoDataset na construção e validação de modelos preditivos de estimativa de esforço; Eles compararam e analisaram o desempenho da previsão de SP com base em recursos de PLN e modelos de AM. Os resultados indicaram que previsões com um modelo *ensemble* (média entre *XGBoost*, *LightGBM* e *CatBoost*) combinando incorporações TF-IDF e SBERT obtiveram um melhor desempenho (ou seja, menor erro em comparação com os seus baselines).
- Kulyabin et al. [74] utilizaram o NeoDataset como ponto de comparação e motivação para a criação de um novo conjunto de dados. NeoDataset não foi usado diretamente para as tarefas de PLN e AM, mas sim como ponto de comparação e motivação para a criação do novo conjunto de dados pelos autores.

Esses trabalhos reforçam a relevância e a aplicabilidade dos artefatos produzidos na Tese, tanto no avanço de soluções de PLN e aprendizado de máquina para engenharia de software quanto na criação de pesquisas futuras.

1.9 Estrutura da Tese

O capítulo 2 apresenta alguns conceitos relacionados à Tese. O capítulo 3 discute trabalhos relacionados. O capítulo 4 apresenta a metodologia e detalhes de suas etapas. O capítulo 5 apresenta o conjunto de dados produzido. Os capítulos 6, 7 e 8 respondem às seguintes questões (QP 2 e QP 3). O capítulo 6 apresenta a *User Story* Tutor para estimar e recomendar melhorias na *User Story*. O capítulo 7 apresenta o modelo preditivo *Effort Model* e os resultados comparativos com os dois *baselines* selecionados. O capítulo 8 descreve a avaliação do *Neo LLM preditor* na previsão da estimativa de esforço em comparação com os *baselines* few-shot e zero-shot. Por fim, o capítulo 9 encerra com as conclusões, discute as limitações encontradas e sugere trabalhos futuros.

Capítulo 2

Fundamentação Teórica

Podemos ver apenas uma curta distância à frente, mas pode-se ver muito que precisa ser feito.

Alan Turing

2.1 Resumo

O objetivo deste capítulo é apresentar os conceitos necessários para o entendimento das outras partes do estudo. Ele apresenta os principais fundamentos teóricos que sustentam a pesquisa, abrangendo áreas interdisciplinares como Engenharia de Software, Processamento de Linguagem Natural (PLN) e Aprendizado de Máquina. São discutidos os conceitos centrais de estimativas de esforço em ambientes ágeis, com ênfase no uso de *Story Points* e *User Stories*. No campo de PLN, detalham-se técnicas de pré-processamento, vetorização textual (como TF-IDF e word embeddings), análise de sentimentos e legibilidade. Também são exploradas abordagens de aprendizado supervisionado e não supervisionado aplicadas à previsão de esforço, além da utilização de Modelos de Linguagem em Grande Escala (LLMs) e suas estratégias de inferência (zero-shot, few-shot e fine-tuning).

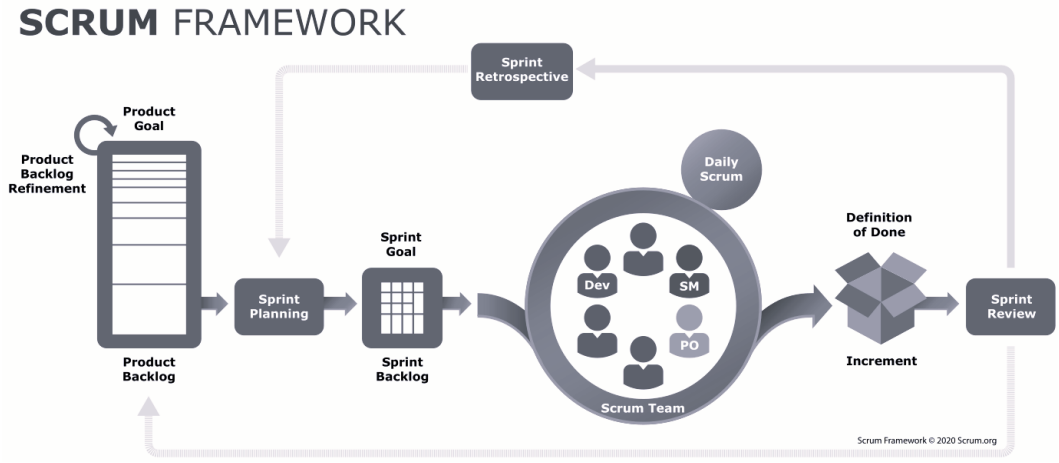
2.2 User Stories

Em 1993, o engenheiro de software Jeff Sutherland criou um processo de desenvolvimento chamado SCRUM [142]. Jeff trabalhava na empresa Easel Corporation e aplicou o SCRUM no projeto mais crítico dessa organização, obtendo bons resultados [124]. Em meados de 2001, ele, Ken Schwaber, Kent Beck e outros 16 pesquisadores discutiram as dificuldades no processo de desenvolvimento tradicional. No final deste encontro, eles elaboraram o manifesto ágil [13]. Esse manifesto deu início ao movimento ágil, do qual o SCRUM faz parte.

O SCRUM é um método ágil baseado em ciclos de tempo fixo, chamados de sprints. Em uma sprint, o time trabalha para atingir objetivos bem definidos. Esses objetivos são registrados em uma lista de coisas a fazer que é constantemente atualizada e re-priorizada, chamada Product Backlog [142].

A característica do SCRUM é o foco em entregas rápidas com alto valor agregado em curtos períodos, lidando com as mudanças o mais rápido possível [39]. Esta abordagem tem mostrado resultados e tem sido utilizada em gerenciamento de projetos [110], tanto na indústria [158, 122] quanto no governo [84]. Vide Figura 2.1.

Figura 2.1: SCRUM Framework



As *User Stories* são peças centrais no registro de requisitos em equipes que usam o SCRUM. Essa narrativa deve ser concisa e descrever a funcionalidade desejada por um usuário em um sistema [27]. Elas podem ser descritas em diferentes

níveis de granularidade, desde funcionalidades abrangentes até detalhes específicos da interface. Também podem ser estruturadas de diversas maneiras, porém, uma estrutura formal comumente utilizada é a de Cohn [27]:

> Como **tipo de usuário (quem)**, eu quero **funcionalidade (o que)**, para que **benefício (porque)**.

Ela é composta por três elementos principais:

- **Quem:** O tipo de usuário que necessita da funcionalidade.
- **O que:** A funcionalidade específica que o usuário precisa.
- **Porque:** A razão pela qual o usuário precisa da funcionalidade e os benefícios que ela trará.

A empresa Mountain Goat Software fornece 200 exemplos de *User Stories* que podem ser utilizados para treinamentos ou estudos. Essas *User Stories* foram escritas durante a construção do site da Scrum Alliance entre 2003 e 2004. Elas estão disponíveis para download em formato PDF [136]. A Tabela 2.1 apresenta alguns exemplos desse conjunto de dados.

Tabela 2.1: Exemplos de *User Stories* do site Mountain Goat Software.

Descrição
Como membro do site, quero descrever-me na minha própria página de uma forma semiestruturada para que os outros possam aprender sobre mim. Ou seja, posso preencher campos predefinidos, mas também ter espaço para um ou dois campos de texto livre. (Seria bom deixar esse texto livre ser Markdown ou similar para permitir alguma formatação simples.)
Como membro do site, posso preencher uma inscrição para me tornar um Certified Scrum Practitioner para que eu possa ganhar essa designação. [Nota: Certified Scrum Practitioner foi o nome inicial do que ficou conhecido como Certified Scrum Professional.]
Como Practitioner, quero que minha página de perfil inclua detalhes adicionais sobre mim (ou seja, algumas das respostas à minha inscrição no Practitioner) para que eu possa mostrar minha experiência.

Outros conjuntos de dados com *User Stories* podem ser encontrados no site Mendeley Data. Esse site é um repositório que armazena vários conjuntos de dados disponíveis para pesquisa. Um dos conjuntos de dados existentes neste repositório

consolida *User Stories* de 22 projetos [30]. A Tabela 2.2 apresenta um exemplo de *User Stories* deste conjunto de dados.

Tabela 2.2: Outro Conjunto de dados com várias *User Stories* ddo Mendeley Data.

Descrição
Como designer de interface do usuário, desejo reprojetar a página Recursos para que ela corresponda aos novos estilos de design do Broker.
Como um designer de interface do usuário, quero relatar às agências sobre testes de usuários, para que eles estejam cientes de suas contribuições para tornar o Broker uma UX melhor.
Como designer de interface do usuário, quero passar para a rodada 2 de edições de página de destino DABS ou FABS, para que eu possa obter aprovações da liderança.

2.3 GitLab

A maioria das equipes que utilizam *User Stories* também utilizam ferramentas de software para gerenciar o projeto e, principalmente, para manter um registro de suas *User Stories* [63]. Ao analisar os dados registrados por essas ferramentas, pode-se extrair informações para diversas pesquisas de engenharia de software, incluindo pesquisas sobre como melhorá-las [64].

O GitLab [48] é uma das ferramentas de gerenciamento utilizadas por equipes ágeis para registrar *User Stories* [25]. Ela permite que os engenheiros de software automatizem muitas ações durante o ciclo de desenvolvimento, incluindo registrar e alterar *User Stories* [36]. No GitLab [48], a *User Story* é registrada como um Issue, e para cada *User Story* são armazenadas diversas informações, como o título, a descrição e sua estimativa em Story Points. Outra ferramenta utilizada com funcionalidades semelhantes é o Jira da empresa Atlassian [9].

Esses dados vêm sendo utilizados em diversas pesquisas relacionadas a problemas em engenharia de software, tais como, atribuição de *User Stories* [80], melhoria da descrição de *User Stories* [19], planejamento da Sprint [22], análise de sentimento de desenvolvedores que escrevem as *User Stories* [103, 104, 165], estimativa de esforço de *User Stories* [111, 135, 37, 23, 150] e priorização de *User*

Stories [47, 58, 162].

O GitLab [48] (em 2023) está localizado no canto superior direito do quadrante Gartner como uma ferramenta líder em DevOps (Figura 2.2). Os líderes do Quadrante Mágico das Plataformas de DevOps de 2023 influenciam a direção do espaço de DevOps com sua liderança de pensamento e serviços. Eles representam plataformas funcionalmente ricas com vários recursos e suportam o software durante todo o ciclo de vida de desenvolvimento.

Figura 2.2: Quadrante mágico Gartner - GitLab

2.4 Índices de Legibilidade

Os índices de Legibilidade, também chamados de índices de leiturabilidade, de apreensibilidade ou, ainda, de facilidade de leitura, são métricas desenvolvidas para avaliar o grau de dificuldade de leitura de um texto [5]. Nada mais são do que alguns valores numéricos extraídos do texto, conhecidos como índices de legibilidade, os

quais são utilizados para avaliar a facilidade de leitura dos textos.

Eles têm sido utilizados na detecção de notícias falsas e spams [24]. Além disso, esses textos podem ser interpretados como um indicador numérico que revela quão fácil é para outras pessoas lerem um determinado texto [38]. Geralmente, o cálculo desses índices leva em consideração informações básicas extraídas do texto, como a quantidade de palavras, caracteres, sentenças, sílabas e listas de palavras complexas. além disso, considera-se o comprimento das frases e a complexidade das palavras.

Esses índices de legibilidade têm sido utilizados por educadores desde 1920 para aprimorar a qualidade do ensino e a compreensão dos alunos. Em 1980, já eram conhecidas 200 fórmulas de cálculo [38]. Esses índices já foram criticados por pesquisadores que apontam suas limitações [72]. Contudo, experimentos empíricos confirmaram a relação entre esses índices e a legibilidade do texto [14].

Cita-se como prova de conceito um software que utiliza estes índices para facilitar a escrita de parágrafos em português: o software ALT [89]. Os autores converteram todas as fórmulas, quando necessário, para a língua portuguesa e disponibilizaram o software para uso gratuito [82]. O ALT já foi utilizado em pelo menos 33 outros trabalhos acadêmicos. A seguir, são apresentadas algumas imagens do resultado da análise de um determinado parágrafo pelo software ALT (Figura 2.3, Figura 2.4 e Figura 2.5).

Figura 2.3: Métricas apresentadas para determinado texto utilizando o ALT.

Métrica	Pontuação
Teste de facilidade de leitura de Flesch	31.4
Índice Gulpease	50.2
Nível de graduação de Flesch-Kincaid	13.5
Índice de nebulosidade de Gunning adaptado	12.4
Índice de legibilidade automatizado (ARI)	13.5
Índice de Coleman-Liau	15.6

Outro software utilizado é o Farfalla. Ele apresenta os índices com um componente visual do tipo velocímetro, que fornece a mesma informação, mas com três fundos diferentes, um para cada nível: elementar, médio e superior [79]. Uma

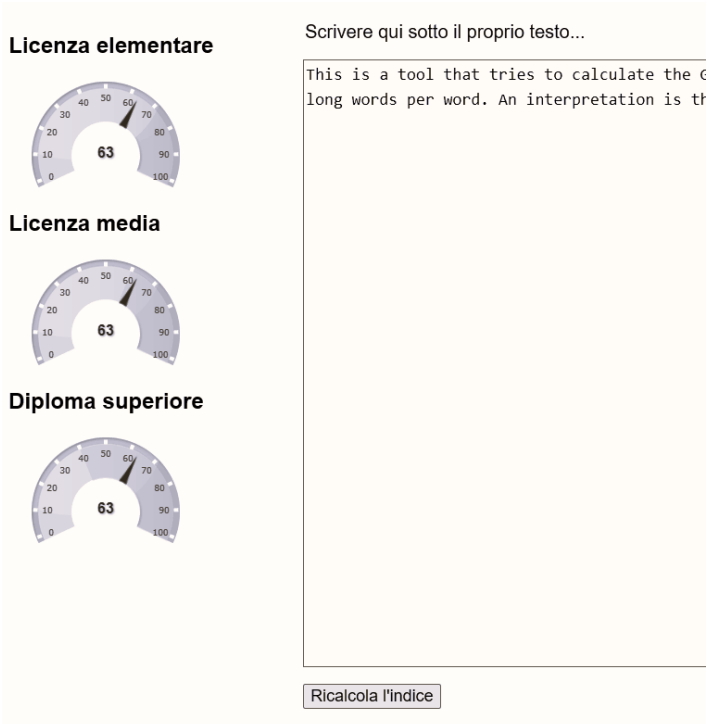
Figura 2.4: Dados extraídos do texto exibidos pelo ALT.

Resumo descritivo	
Letras	464
Sílabas	200
Palavras	81
Sentenças	5
Letras/palavra	5.7
Sílabas/palavra	2.5
Palavras/sentença	16.2
Palavras complexas	19 (23.5%)

Figura 2.5: Resultado Geral a partir de determinado texto exibido pelo ALT.

imagem do resultado é apresentada na Figura 2.6.

Figura 2.6: Análise textual do software farfalla.



A seguir, são descritos alguns dos índices de legibilidade, a saber: *Gunning Fog*,

Flesh Reading Ease, *Coleman Liau Index*, *Linsear Write Formula*, *Dale–Chall readability formula*, *Automated readability index*, além de Subjetividade e Polaridade.

2.4.1 Gunning Fog

O Índice *Gunning Fog* [54] é um dos mais citados em pesquisas cientificas relacionadas a facilidade de leitura do texto [14]. Ele serve para resumir um texto, incorporando nessa informação numérica a dificuldade de leitura do mesmo [14]. Em outras palavras, trata-se de um número atribuído a um determinado texto que considera, em sua fórmula, a quantidade de palavras, a quantidade de frases e uma lista de palavras complexas.

O *Gunning Fog* foi criado por Robert Gunning em 1954. De acordo com a teoria em que se baseia, uma maior porcentagem de palavras complexas torna a leitura do texto mais difícil. Portanto, quanto maior o índice, menor será a legibilidade do texto. O algoritmo e o método de cálculo estão bem documentados [53]. O índice é calculado somando o comprimento médio da frase (a quantidade de palavras dividida pela quantidade de sentenças) à porcentagem de palavras complexas (a quantidade de palavras complexas dividida pela quantidade total de palavras) e multiplicando-se essa soma por 0,4. Como indicado na fórmula apresentada na Equação 2.1.

$$0.4 \times \left[\left(\frac{qtd\ palavras}{qtd\ sentencas} \right) + 100 \times \left(\frac{qtd\ palavras\ complexas}{qtd\ palavras} \right) \right] \tag{2.1}$$

Este índice mede a complexidade de um texto com base na dificuldade de suas palavras e na estrutura das sentenças. Uma parte essencial do cálculo envolve a identificação de palavras complexas (a definição de palavra complexa é discutida na seção 2.4.2) que são utilizadas para estimar o esforço de leitura do texto.

A Tabela 2.3 apresenta as faixas do índice Gunning Fog Index, sua interpretação semântica e o nível de escolaridade formal estimado para a plena compreensão do texto. Esse índice expressa diretamente o número aproximado de *anos de estudo* exigidos, considerando o comprimento das frases e a proporção de palavras complexas.

Tabela 2.3: Faixas do índice Gunning Fog Index e respectivas interpretações

Valor GFI	Classificação	Interpretação semântica	Escolaridade típica
6–8	Muito fácil	Texto simples, com frases curtas e vocabulário cotidiano.	Ensino Fundamental I
9–12	Fácil	Texto claro e acessível ao público geral.	Ensino Fundamental II
13–16	Moderado	Estrutura frasal mais densa e vocabulário elaborado.	Ensino Médio
17–18	Difícil	Texto com alta densidade léxica e termos técnicos frequentes.	Graduação
19–22	Muito difícil	Texto acadêmico complexo, com sentenças longas e palavras especializadas.	Pós-graduação (mestrado)
≥ 23	Extremamente difícil	Texto altamente técnico e conceitualmente denso, típico de produção científica avançada.	Pós-graduação (doutorado / pesquisadores)

Por exemplo, quando um texto apresenta um valor de Gunning Fog Index igual a 23,3, isso indica que são necessários, em média, cerca de 23 anos de escolaridade formal para sua plena compreensão. Semanticamente, esse resultado caracteriza um texto *extremamente difícil*, típico de artigos científicos avançados, teses e outros documentos acadêmicos voltados a leitores especialistas.

Assim como outros índices o Gunning Fog Index não está associado a uma área específica do conhecimento, mas à complexidade linguística do texto, estimada a partir do comprimento médio das frases e da proporção de palavras consideradas complexas.

2.4.2 Palavras complexas

As métricas discutidas nesta seção utilizam diferentes abordagens para avaliar a complexidade das palavras. Os Índices Coleman-Liau utilizam o comprimento médio das palavras, em termos do número de letras, para a análise linguística [5]. Os testes de Flesch-Kincaid utilizam o critério da quantidade de sílabas presentes em cada palavra. Há também métricas que utilizam as frequências das palavras no uso cotidiano para inferir suas complexidades: quanto menos frequente, mais complexa [5].

Uma definição para “palavra complexa”, segundo Gross and Sadowski [53], utilizada no cálculo do Gunning Fog, é que uma *palavra complexa* é aquela que possui um número significativo de sílabas e que não é de uso comum.

- Possui três ou mais sílabas, com exceção de:
  - Palavras próprias (nomes de pessoas, lugares, etc.);

  - Palavras compostas, como *porta-retrato*;
  - Verbos com sufixos comuns em inglês (-ed, -es, -ing);
  - Palavras familiares e de uso comum, mesmo que longas (por exemplo, *important*, *yesterday*).

A seguir, são apresentados alguns exemplos de Palavras Complexas em Inglês. Essas palavras são comumente classificadas como complexas no cálculo do *Gunning Fog Index*.

Tabela 2.4: Exemplos de palavras complexas em inglês

Categoria	**Exemplos**
Acadêmicas	approximately, significant, methodology, interpretation
Técnicas	configuration, implementation, transformation, verification
Abstratas	assumption, evaluation, perception, consideration
Políticas e jurídicas	administration, legislation, constitution, regulation
Gerais longas	opportunity, information, environment, development

Quando aplicado ao português, o conceito é mantido: considera-se *complexa* a palavra com três ou mais sílabas que não seja:

- Palavra de uso muito comum (como *agora*, *então*, *porque*);
- Palavra composta ou com sufixo verbal (como *falando*, *correndo*).

A Tabela 2.5 apresenta uma amostra de dez palavras complexas que são comumente encontradas em textos acadêmicos e técnicos.

Tabela 2.5: Amostra de palavras complexas em português

#	**Palavra**
1	metodologia
2	implementação
3	responsabilidade
4	regulamentação
5	aproximadamente
6	interdisciplinaridade
7	transformação
8	consideração
9	compatibilidade
10	interpretação

2.4.3 Flesch Reading Ease

Um índice de legibilidade de texto é o Flesch Reading Ease, conforme indicado em Textstat [154]. Quanto menor o valor, mais difícil será a leitura do texto. A Equação 2.2 apresenta o cálculo. Este é um dos testes mais antigos e mais utilizados, e depende apenas de dois fatores: Quanto maior o comprimento médio da frase (quantidade de palavras dividido pela quantidade de sentenças), mais difícil se torna a leitura do texto. Quanto maior o número médio de sílabas de uma palavra, mais difícil se torna a leitura do texto. Quanto maior a pontuação, maior será a legibilidade do texto.

$$206.835 - 1.015 \times \left(\frac{qtd\ palavras}{qtd\ sentencas}\right) - 84.6 \times \left(\frac{qtd\ silabas}{qtd\ palavras}\right) \tag{2.2}$$

A Tabela 2.6 apresenta a faixa completa do índice Flesch Reading Ease em português, com a classificação qualitativa, interpretação semântica e o nível de escolaridade típico associado a cada intervalo.

Tabela 2.6: Faixas do índice Flesch Reading Ease e respectivas interpretações

Pontuação Flesch	**Classificação**	**Interpretação semântica**	**Escolaridade típica**
90–100	Muito fácil	Texto extremamente simples, com frases curtas e palavras comuns.	4ª série do Ensino Fundamental
80–89	Fácil	Fácil de ler; adequado ao público geral.	5ª–6ª série
70–79	Relativamente fácil	Leitura confortável para a maioria das pessoas.	7ª série
60–69	Padrão / Normal	Complexidade moderada; típico de jornais.	8ª–9ª série
50–59	Um pouco difícil	Vocabulário mais elaborado; exige maior atenção do leitor.	Ensino Médio
30–49	Difícil	Estruturas densas e termos complexos; leitura acadêmica inicial.	Nível superior (graduação)
0–29	Muito difícil	Texto bastante complexo; típico de artigos acadêmicos avançados.	Pós-graduação (mestrado/doutorado)
< 0	Extremamente difícil	Texto altamente técnico e denso, com léxico sofisticado e frases muito longas.	Especialistas e pesquisadores

Por exemplo, quando a pontuação do Flesch Reading Ease é muito baixa, como no caso de um valor igual a 3,9, o texto situa-se na faixa de 0–29 pontos. Nessa situação, a classificação é *muito difícil*, indicando que apenas leitores com nível de pós-graduação tendem a compreender o texto com fluência. Trata-se de textos com alta complexidade linguística, vocabulário sofisticado e estruturas frasais densas, típicos de artigos acadêmicos avançados.

Ressalta-se que o índice não está associado a uma área específica do conheci-

mento (por exemplo, Computação, Direito ou Medicina), mas sim ao nível de escolaridade e à experiência do leitor com textos complexos.

2.4.4 Coleman Liau Index

Outro índice de legibilidade de texto é o Coleman Liau Index [154]. Este utiliza a Equação 2.3. Onde L representa a média do número de letras a cada 100 palavras, e S representa a média de sentenças a cada 100 palavras.

$$CLI = 0.0588 \times L - 0.296 \times S - 15.8 \tag{2.3}$$

A Tabela 2.7 apresenta as faixas do índice Coleman–Liau Index, sua interpretação semântica e o nível de escolaridade típico associado. Diferentemente de outros índices de legibilidade, o Coleman–Liau baseia-se no comprimento médio das palavras (em número de caracteres) e no comprimento médio das sentenças, não considerando sílabas.

Tabela 2.7: Faixas do índice Coleman–Liau Index e respectivas interpretações

Valor CLI	Classificação	Interpretação semântica	Escolaridade típica
1–4	Muito básico	Texto extremamente simples, com palavras curtas e frases diretas.	Ensino Fundamental I
5–6	Básico	Texto claro, com baixa complexidade lexical e sintática.	Ensino Fundamental II (inicial)
7–8	Intermediário	Estrutura textual moderada, adequada a leitores em consolidação.	Ensino Fundamental II (final)
9–10	Padrão	Complexidade linguística típica de textos informativos.	Ensino Médio
11–12	Avançado	Vocabulário mais elaborado e sentenças mais longas.	Graduação
$\geq$ 13	Muito avançado	Texto denso, com léxico sofisticado e alta complexidade sintática.	Pós-graduação

Por exemplo, quando um texto apresenta um valor de Coleman–Liau Index igual a 10,4, isso indica que o texto exige, aproximadamente, o equivalente ao final do Ensino Médio ou início da graduação para sua compreensão. Semanticamente, trata-se de um texto de complexidade *padrão a avançada*, caracterizado por palavras mais longas e sentenças com maior densidade informacional.

O Coleman-Liau Index, assim como outros índices, não se associa a uma área específica do conhecimento, mas reflete exclusivamente a complexidade linguística do texto, estimada a partir do comprimento médio das palavras e das sentenças.

2.4.5 Automated readability index

Outro índice de legibilidade de texto é o *Automated readability index* a partir da Equação 2.4, conforme [154].

$$4.71 \times \left(\frac{qtd\ caracteres}{qtd\ palavras} \right) + 0.5 \times \left(\frac{qtd\ palavras}{qtd\ sentencas} \right) - 21.43 \tag{2.4}$$

A Tabela 2.8 apresenta as faixas do *Automated Readability Index* (ARI), sua interpretação semântica e o nível de escolaridade típico associado. O ARI estima a legibilidade do texto com base no comprimento médio das palavras (em caracteres) e no comprimento médio das sentenças, fornecendo diretamente uma estimativa do nível educacional requerido.

Tabela 2.8: Faixas do índice Automated Readability Index (ARI) e respectivas interpretações

Valor ARI	Classificação	Interpretação semântica	Escolaridade típica
1–4	Muito básico	Texto simples, com palavras curtas e sentenças diretas.	Ensino Fundamental I
5–7	Básico	Texto claro, com baixa complexidade estrutural.	Ensino Fundamental II
8–10	Intermediário	Estrutura frasal mais elaborada e vocabulário moderado.	Ensino Médio
11–12	Avançado	Texto com maior densidade informacional e vocabulário elaborado.	Leitores experientes / final do Ensino Médio
13–16	Muito avançado	Texto acadêmico denso, com sentenças longas e termos técnicos.	Graduação
≥ 17	Extremamente avançado	Texto altamente técnico e conceitualmente complexo.	Pós-graduação

Quando um texto apresenta um valor de Automated Readability Index igual a 11,6, isso indica que o texto é adequado para leitores experientes, com domínio consolidado da leitura e capacidade de compreender estruturas frasais mais longas e vocabulário mais elaborado. Em termos educacionais, esse valor situa o texto no limite superior do Ensino Médio ou início do nível superior.

O ARI, assim como os demais índices de legibilidade apresentados, não está associado a uma área específica do conhecimento, mas reflete a complexidade linguística do texto, estimada a partir do comprimento médio das palavras e das sentenças.

2.4.6 Flesch-Kincaid Grade Level

Outro índice de legibilidade de texto é o *Flesch-Kincaid Grade* do texto, conforme [154]. Veja a Equação 2.5.

$$0.39 \times \left(\frac{qtd\ palavras}{qtd\ sentencas} \right) + 11.8 \times \left(\frac{qtd\ silabas}{qtd\ palavras} \right) - 15.59 \tag{2.5}$$

A Tabela 2.9 apresenta as faixas do índice Flesch–Kincaid Grade Level, sua classificação semântica e o nível de escolaridade típico associado. Diferentemente do Flesch Reading Ease, este índice expressa diretamente o *ano de escolaridade* necessário para a compreensão do texto.

Tabela 2.9: Faixas do índice Flesch–Kincaid Grade Level e respectivas interpretações

Valor FKGL	Classificação	Interpretação semântica	Escolaridade típica
1–4	Muito básico	Texto extremamente simples, com frases curtas e vocabulário elementar.	Ensino Fundamental I
5–8	Básico	Texto claro e direto, adequado a leitores em formação.	Ensino Fundamental II
9–12	Intermediário	Estrutura frasal mais elaborada; leitura escolar avançada.	Ensino Médio
13–16	Avançado	Vocabulário técnico moderado e maior densidade sintática.	Graduação
17–18	Muito avançado	Texto acadêmico denso, com termos especializados.	Pós-graduação (mestrado)
≥ 19	Extremamente avançado	Texto altamente complexo, técnico e conceitualmente denso.	Pós-graduação (doutorado / pesquisadores)

Por exemplo, Quando um texto apresenta um valor de Flesch–Kincaid Grade Level igual a 19,3, isso indica que seriam necessários, em média, mais de 19 anos de escolarização formal para compreendê-lo adequadamente. Tal valor corresponde semanticamente a um *nível de pós-graduação*, sendo típico de textos acadêmicos avançados, como artigos científicos, teses e dissertações.

Assim como ocorre com o Flesch Reading Ease, o índice Flesch–Kincaid Grade Level não está associado a uma área específica do conhecimento, mas sim ao nível educacional e à complexidade linguística do texto, medida a partir do comprimento das frases e do número médio de sílabas por palavra.

2.4.7 Linsear Write Formula

Outro índice de legibilidade de texto é o *Linsear Write Formula* [154]. A métrica Lw requer um texto contendo, no mínimo, 100 palavras.

O algoritmo funciona da seguinte forma:

- Para cada palavra mais simples, definido como palavras que tem somente 2 sílabas ou menos, adicione um ponto;
- Para cada palavra mais difícil, definida como uma palavra de 3 sílabas ou mais, adicione 3 pontos;
- Divida os pontos pelo número de sentenças na amostra de 100 palavras;
- Ajuste o valor temporário de r:
  - Se $r > 20$ então $Lw = r/2$
  - Senão $r < 20$ então $Lw = r/2 - 1$

A Tabela 2.10 apresenta as faixas do *Linsear Write Formula*, sua interpretação semântica e o nível de escolaridade típico associado. Esse índice foi originalmente desenvolvido para avaliar textos técnicos, considerando a proporção de palavras simples e complexas e a estrutura sintática das sentenças.

Tabela 2.10: Faixas do índice Linsear Write Formula e respectivas interpretações

Valor LWF	**Classificação**	**Interpretação semântica**	**Escolaridade típica**
1–4	Muito simples	Texto elementar, com vocabulário básico e frases curtas.	Ensino Fundamental I
5–8	Simples	Texto acessível, com baixa complexidade sintática.	Ensino Fundamental II
9–12	Intermediário	Estrutura frasal mais elaborada e vocabulário moderado.	Ensino Médio
13–16	Difícil	Alta densidade informacional e uso frequente de termos técnicos.	Graduação
17–18	Muito difícil	Texto acadêmico denso, com sentenças longas e léxico especializado.	Pós-graduação (mestrado)
≥ 19	Extremamente difícil	Alta complexidade sintática e lexical, típica de textos científicos avançados.	Pós-graduação (doutorado / especialistas)

Por exemplo, quando um texto apresenta um valor de Linsear Write igual a 19,0, isso confirma a presença de alta complexidade sintática e lexical. Tal pontuação é característica de textos científicos e técnicos avançados, nos quais predominam sentenças longas, vocabulário especializado e alta densidade informacional.

O Linsear Write Formula, assim como os demais índices de legibilidade apresentados, não se associa a uma área específica do conhecimento, mas reflete a complexidade linguística do texto, sendo particularmente adequado para avaliar documentos técnicos e acadêmicos.

2.4.8 Dale–Chall readability formula (ou Dale–Chall Score)

Outro índice de legibilidade de texto é o *Dale–Chall readability formula*. Calcula-se a legibilidade com base no nível de vocabulário e no comprimento das frases. A fórmula é comumente utilizada para determinar a dificuldade de textos em diversos contextos, como materiais educacionais e documentos online. Ela utiliza uma tabela de *lookup table* as 3.000 palavras em inglês mais utilizadas [154]. A Equação 2.6 apresenta esse cálculo.

$$0.1579 \times \left(\frac{qtd\ palavras\ dificeis}{qtd\ total\ palavras} \times 100 \right) + 0.0496 \times \left(\frac{qtd\ total\ palavras}{qtd\ total\ sentencas} \right) \quad (2.6)$$

Vários estudos já utilizaram a fórmula de Dale-Chall para medir a legibilidade, incluindo componentes de literatura para alunos do ensino médio [143, 51]. A fórmula é utilizada para comparar a legibilidade de diferentes textos e avaliar a qualidade dos documentos em serviços de perguntas e respostas [91].

A Tabela 2.11 apresenta as faixas do *Dale–Chall Readability Score*, sua interpretação semântica e o nível de escolaridade típico associado. Diferentemente de outros índices de legibilidade, o Dale–Chall baseia-se na proporção de palavras consideradas familiares ao leitor médio, sendo sensível ao uso de vocabulário técnico ou especializado.

Tabela 2.11: Faixas do índice Dale–Chall Readability Score e respectivas interpretações

Valor DC	Classificação	Interpretação semântica	Escolaridade típica
4,9 ou menos	Muito fácil	Vocabulário amplamente familiar, frases simples.	Ensino Fundamental I
5,0–5,9	Fácil	Texto claro, com poucos termos não familiares.	Ensino Fundamental II
6,0–6,9	Moderado	Vocabulário parcialmente técnico; exige maior atenção.	Ensino Médio
7,0–7,9	Difícil	Uso frequente de termos menos familiares ao leitor médio.	Graduação
8,0–8,9	Muito difícil	Vocabulário técnico e especializado predominante.	Pós-graduação (mestrado)
9,0–9,9	Extremamente difícil	Alta densidade de termos técnicos e conceitos abstratos.	Pós-graduação (doutorado)
≥ 10	Altamente técnico	Vocabulário altamente especializado, típico de literatura científica.	Especialistas e pesquisadores

Quando um texto apresenta um valor de Dale–Chall Score igual a 11,9, isso indica uma elevada proporção de palavras não familiares ao leitor médio, sinalizando

o uso intensivo de termos técnicos e vocabulário especializado. Semanticamente, tal pontuação caracteriza um texto *altamente técnico*, voltado a leitores especialistas, com formação avançada e domínio do jargão da área.

O Dale–Chall Score não se associa a uma área específica do conhecimento, mas reflete diretamente o grau de especialização do vocabulário empregado no texto, sendo particularmente sensível à presença de termos técnicos e científicos.

2.5 Estimativa de Esforço

No campo do desenvolvimento de software, a estimativa de esforço envolve a tarefa de prever a quantidade mais precisa de esforço necessária para a criação ou manutenção do software. Geralmente, essa estimativa é realizada com dados incompletos, incertos ou ruidosos. Essas estimativas de esforço podem servir como insumos valiosos para planos de projetos, planos de iteração, orçamentos financeiros, avaliações de investimentos e estratégias de montagem de preços em contratos e licitações [170, 61, 101].

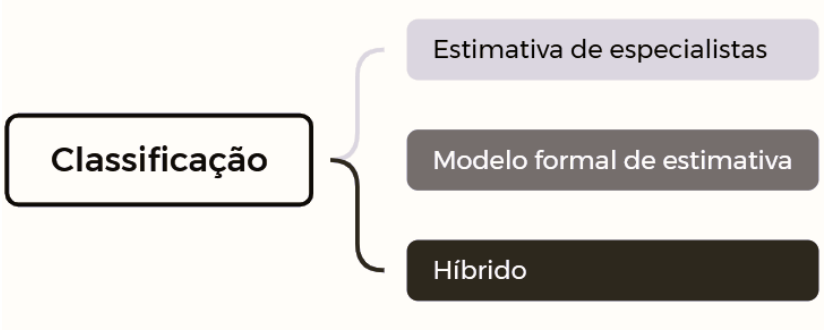
Existem vários fatores que influenciam a tarefa de estimar o esforço no desenvolvimento de software [73]. Um desses fatores é a falta de informação sobre o passado, incluindo o conhecimento e a experiência dos especialistas [12]. Além disso, os dados históricos do projeto frequentemente apresentam lacunas. Outro problema é que, na maioria das vezes, esses modelos são baseados em pessoas, o que os torna não repetíveis e suscetíveis a mudanças nos requisitos, bem como nas ferramentas e tecnologias utilizadas. Entretanto, apesar de sua complexidade, essas estimativas de esforço funcionam como um ponto de partida para diversas atividades de gerenciamento de projetos; portanto, é crucial obter as estimativas mais precisas possíveis [109].

2.5.1 Classificação dos métodos de estimativa

Inúmeros métodos de estimativa têm sido propostos pelos pesquisadores para resolver o problema da estimativa de esforço [163]. As abordagens podem ser

agrupadas de diversas maneiras [170, 16, 66]. Uma dessas classificações é apresentada a seguir (Figura 2.7).

Figura 2.7: Classificação dos métodos de estimativa de esforço.



- A estimativa de especialistas refere-se à etapa de quantificação, na qual a estimativa é produzida com base no julgamento individual desses especialistas.
- Modelo formal de estimativa: A etapa de quantificação baseia-se em algoritmos ou fórmulas, utilizando uma fórmula derivada de dados históricos.
- Estimativa híbrida: A etapa de quantificação baseia-se em uma combinação dos dois elementos anteriores.

Pesquisas publicadas sobre práticas de estimativa sugerem que a estimativa de especialistas é a estratégia dominante ao estimar o esforço de desenvolvimento de software [170, 65].

2.5.2 Aprendizagem de Máquina na estimativa

A inteligência artificial (IA) e a aprendizagem de máquina (AM) têm sido utilizadas para solucionar diversos problemas relacionados a engenharia de software, como a atribuição de *User Stories* [80], o aprimoramento da descrição de *User Stories* [19], o planejamento das iterações de times ágeis [22], a análise do sentimento dos desenvolvedores que escrevem as *User Stories* [103, 104, 165], a estimativa de esforço para *User Stories* [111, 135, 37, 23, 147] e a priorização de *User Stories* [47, 58, 162]. Por fim, Dantas et al. [33] utilizaram o algoritmo Árvore de Decisão, que teve desempenho superior em comparação com a estimativa da equipe, confirmando que o uso de AM pode auxiliar no processo de estimativa.

2.5.3 Vantagens e desvantagens da AM na estimativa

A Tabela 2.12 apresenta as vantagens e desvantagens do uso de AM na estimativa de esforço. Embora o aprendizado de máquina ofereça vantagens significativas em termos de precisão e eficiência na estimativa de esforço, também apresenta desafios relacionados à complexidade, à qualidade dos dados, à interpretabilidade e ao uso intensivo de recursos.

Tabela 2.12: Vantagens e desvantagens do uso de AM na estimativa.

Vantagem / Desvantagem	**Descrição**	**Detalhamento**
Vantagens	Precisão e previsões mais aprimoradas	As técnicas de aprendizado de máquina oferecem estimativas de esforço mais precisas em comparação com métodos tradicionais, como julgamento de especialistas e estimativa algorítmica [159, 121].
Vantagens	Adaptabilidade	Algoritmos de aprendizado de máquina podem se adaptar às mudanças nos requisitos e dados do projeto, melhorando a estimativa ao longo do tempo [8].
Vantagens	Eficiência	Essas técnicas podem agilizar o processo de estimativa de esforço, economizando tempo e recursos em projetos de desenvolvimento de software [55].
Desvantagens	Complexidade	A implementação de algoritmos de aprendizado de máquina para estimativa de esforço pode, por exemplo, exigir conhecimentos e habilidades especializadas, acrescentando complexidade ao processo de estimativa [55].
Desvantagens	Dependência de dados	Os modelos de aprendizado de máquina dependem fortemente de dados de qualidade, e dados imprecisos ou insuficientes podem levar a estimativas erradas [121].
Desvantagens	Interpretabilidade	Alguns modelos de aprendizagem automática carecem de transparência na forma como chegam às estimativas, dificultando compreender o raciocínio por detrás das previsões [8].
Desvantagens	Uso intensivo de recursos	o treinamento de modelos de aprendizado de máquina pode ser computacionalmente intensivo e pode exigir recursos computacionais significativos [117].

2.5.4 Desafios do uso de AM na estimativa

A estimativa de esforço em sistemas de software constitui um trabalho desafiador para gerentes de projetos. Esse desafio surge devido a diversos fatores, incluindo o fator humano, a complexidade do produto, diferentes métodos de estimativa e outros aspectos [90]. Além disso, a implementação de AM na estimativa apresenta novos desafios (Tabela 2.13). Lidar com esses desafios é crucial para a implementação bem-sucedida de técnicas de AM na estimativa de esforço em

projetos de desenvolvimento de software.

Tabela 2.13: Desafios de AM na estimativa.

Descrição	Detalhamento
Qualidade dos dados	Na estimativa de esforço, dados estimados pelo time com falhas podem criar modelos imprecisos e inadequados.
Interpretabilidade	Alguns modelos do tipo caixa preta não apresentam como chegam às estimativas. Modelos mais interpretáveis podem não ser os melhores, porém podem auxiliar melhor os times em relação a quais fatores influenciam mais a estimativa.
Seleção de algoritmos	Escolher os algoritmos e modelos mais adequados para um a estimativa de esforço é um desafio devido à alteração do resultado dadas as características particulares dos dados.
Recursos intensivos	O treinamento de modelos de AM para estimativa de esforço pode ser computacionalmente intensivo e exigir recursos computacionais significativos.
Comunicação e Colaboração	Estimativa pode depender de informações de várias pessoas, como o dono do projeto, o time de desenvolvimento e partes interessadas de negócios, o que pode ser desafiador devido a diferentes perspectivas e experiências e falhas na comunicação.
Dependência de dados	A precisão dos modelos é altamente dependente da qualidade, quantidade e diversidade dos dados disponíveis, o que pode representar desafios, pois os projetos podem ter características distintas.

2.5.5 Métricas de avaliação

Diversas medidas são utilizadas na literatura sobre estimativa de esforço de software para avaliar a precisão dos modelos de estimativa. Essas medidas geralmente utilizam, em seu cálculo, o valor previsto e o valor real [150]. Algumas das técnicas utilizadas neste estudo são apresentadas nesta seção, incluindo o MAE, o MMRE e o MdMRE.

MAE

O *Mean Absolute Error* (MAE) já foi utilizado em outros estudos [132, 76, 127, 150]; portanto, essa medida é adotada como parâmetro para a avaliação dos modelos.

O MAE é calculado conforme a Equação 2.7. Onde o E_{atual} representa o esforço atual, o $E_{predito}$ indica o esforço predito e o n corresponde ao número de observações. Onde i representa a enésima observação, conforme mencionado em i=1, 2 ... n.

$$MAE = \frac{1}{n}\sum_{i=1}^{n}\left|E_{atual_i} - E_{predito_i}\right| \tag{2.7}$$

MMRE

Outro critério de avaliação é a Média da Magnitude Relativa do Erro (MMRE) [45, 75]. A Magnitude Relativa do Erro (MRE) pode ser definida de acordo com a Equação 2.8. Onde E_{atual} representa o esforço atual e $E_{predito}$ refere-se ao esforço predito.

$$MRE = \frac{\left|E_{atual} - E_{predito}\right|}{E_{atual}} \tag{2.8}$$

A Média da Magnitude Relativa do Erro (MMRE) é definida como a média do MRE. O MMRE pode ser definido pela Equação 2.9. Onde n representa o número de observações e MRE_i indica a magnitude do erro da i_{th} observação.

$$MMRE = \frac{1}{n}\sum_{i=1}^{n} MRE_i \tag{2.9}$$

MdMRE

Outro critério de avaliação é a Mediana de Magnitude de Erros Relativos (MdMRE), conforme definido na Equação 2.10.

$$MdMRE = median(MRE_i) \tag{2.10}$$

2.5.6 Instabilidade da Conclusão

A *instabilidade da conclusão* (*conclusion instability*) é um problema recorrente em modelos de estimativa de esforço baseados em aprendizado de máquina. Esse fenômeno caracteriza-se pela variação das conclusões obtidas a partir de pequenas alterações nos dados de entrada, nos hiperparâmetros ou nas partições de treinamento e teste. Essa instabilidade faz com que diferentes execuções do modelo produzam resultados divergentes quanto aos atributos mais relevantes para a

estimativa ou aos valores previstos de esforço [69].

Na prática, tal fenômeno ocorre devido a fatores como o tamanho limitado dos conjuntos de dados, a presença de ruído e subjetividade nas medições de esforço, a alta correlação entre variáveis independentes e o sobreajuste (overfitting) de modelos complexos. Como consequência, o algoritmo pode indicar, em diferentes execuções, atributos distintos como os mais influentes [15].

A falta de consistência afeta diretamente a confiabilidade científica e a interpretação prática dos modelos, pois as partes interessadas podem extrair conclusões contraditórias sobre os fatores determinantes do esforço. Além disso, a instabilidade compromete a reprodutibilidade dos estudos e a comparabilidade entre as diferentes abordagens publicadas na literatura [83].

Entre as estratégias para mitigar esse problema, destacam-se: (i) a utilização de validação cruzada repetida para avaliar a variabilidade das métricas entre execuções; (ii) o uso de métodos de explicabilidade mais robustos; (iii) a adoção de modelos de *ensemble*, que reduzem a variância das previsões individuais; e (iv) a condução de estudos de replicação em diferentes bases de dados, assegurando maior generalização dos resultados [113].

Portanto, reconhecer e tratar a instabilidade da conclusão é essencial para aumentar a confiabilidade dos modelos preditivos de esforço, especialmente em contextos empíricos que envolvem dados heterogêneos, subjetivos e suscetíveis a ruído.

2.6 Processamento de linguagem natural

O Processamento de linguagem natural (PLN) é um campo que utiliza técnicas computacionais com o propósito de aprender, compreender e produzir conteúdo em linguagem humana [174]. Os sistemas de PLN podem suportar diferentes níveis ou combinações de níveis de análise linguística [174]. Os níveis de análise linguística referem-se à análise fonética, morfológica, léxica, sintática, semântica, de discurso e à análise pragmática da linguagem; existe a suposição de que os seres humanos normalmente utilizam todos esses níveis para produzir ou compreender a lingua-

gem [174].

As abordagens de PLN podem ser classificadas em dois grandes grupos: o PLN simbólico e o PLN estatístico [174]. Embora ambos os tipos de PLN tenham sido investigados simultaneamente, foi o PLN simbólico que dominou o campo por um período significativo. Entretanto, as abordagens estatísticas ganharam força principalmente após a divulgação do ChatGPT [102].

2.6.1 Large Language Models

Um Modelo de Linguagem de Larga Escala - um Large Language Models em inglês (LLM), pode ser definido como um sistema computacional fundamentado em técnicas de aprendizado de máquina, cuja finalidade é gerar texto a partir de uma sequência textual fornecida como entrada. Nesse contexto, o texto de entrada é denominado *prompt*, também referido em inglês como *input*, enquanto o texto produzido pelo modelo é denominado resposta ou *output*. A Figura 2.8 apresenta uma representação esquemática do processo de entrada e saída de texto em um LLM.

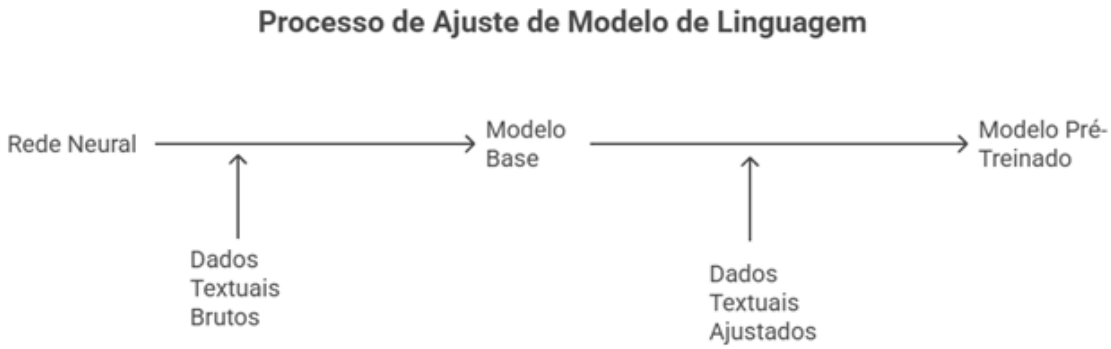
Figura 2.8: Fluxo entrada e saída do LLM

Um LLM é um tipo específico de rede neural artificial, caracterizada por ser uma rede neural profunda com uma arquitetura particular, treinada com dados textuais. Em outras palavras, trata-se de uma grande rede neural que foi treinada com uma quantidade significativa de dados. O conceito de “grande rede neural” está relacionado à quantidade de parâmetros que compõem a rede, a qual pode variar na casa dos milhões; em relação a “muitos dados”, o termo refere-se a dezenas de gi-

gabytes ou mais, abrangendo conjuntos de dados brutos que incluem livros, artigos, páginas da web, documentos técnicos e outros materiais escritos. Esses conjuntos de dados geralmente não cabem totalmente na memória da máquina em que são treinados.

O processo de treino envolve duas etapas (veja a Figura 2.9): (i) um treinamento com dados textuais brutos e (ii) um treinamento com dados textuais anotados, que são mais adequados às necessidades de um LLM [118].

Figura 2.9: Alto nível do processo de treino de um modelo de LLM genérico



Quando um LLM é treinado com dados brutos, ele é denominado modelo “base” e tem como objetivo prever a próxima sequência de palavras a partir do prompt de entrada. Por exemplo, para o prompt: O “livro está”, a resposta de um modelo base poderia ser: “O livro está em cima da mesa.”. A seguir, apresenta-se, no Exemplo 1, um conjunto ilustrativo de dados textuais de treino [118]. Este prompt serve para ilustrar os artefatos de entrada no pré-processamento dos artefatos de entrada utilizados no ajuste fino.

```
Exemplo 1

"Algum tempo hesitei se devia abrir estas memórias
pelo princípio ou pelo fim, isto é, se poria em
primeiro lugar o meu nascimento ou a minha morte.
Suposto o uso vulgar seja começar pelo nascimento,
duas considerações me levaram a adotar diferente
```

método: a primeira é que eu não sou propriamente um autor defunto, mas um defunto autor, para quem a campa foi outro berço; a segunda é que o escrito ficaria assim mais galante e mais novo. Moisés, que também contou a sua morte, não a pôs no intróito, mas no cabo: diferença radical entre este livro e o Pentateuco."
Trecho de Assis, Machado. Todos os Romances: Machado de Assis (Portuguese Edition) (p. 226). Edição do Kindle.

Na segunda etapa, o modelo base é treinado novamente, mas agora com dados de texto anotados por humanos, permitindo que o modelo aprenda não apenas padrões linguísticos gerais, mas também instruções explícitas de interação. Veja o exemplo de dados textuais de treino apresentado no Exemplo 2.

Exemplo 2

user: "Por que um autor começaria sua história pela própria morte, como em Memórias Póstumas?"
bot: "Que boa pergunta. Porque inverter a ordem tradicional cria efeito irônico e destaca a perspectiva do narrador após a morte. Você quer saber mais sobre este assunto?"

user: "Explique o sentido metafórico de 'um defunto autor'."
bot: "Claro. O sentido é a ideia de um narrador que escreve já depois de morto, simbolizando distanciamento crítico e liberdade para julgar a própria vida. Gostaria de saber mais alguma coisa?"

Os Exemplos 1 e 2 ilustram, de maneira simplificada, como trechos textuais podem ser utilizados no treinamento de um modelo de linguagem. Em aplicações reais, o processo envolve outras etapas e conjuntos massivos de dados textuais,

frequentemente compostos por bilhões de palavras provenientes de diversas fontes e organizados com formatações específicas. A execução desse treinamento em larga escala só se tornou possível a partir dos avanços na arquitetura da rede, culminando no Transformer, que introduziu um mecanismo de atenção capaz de lidar, de forma eficiente e paralela, com dependências de longo alcance em sequências de texto.

Modelos de linguagem de larga escala são treinados para prever a próxima palavra em uma sequência, uma habilidade que serve de base para a construção de representações linguísticas sofisticadas. Esse processo permite que modelos compreendam contextos extensos e realizem tarefas complexas de processamento de linguagem natural, como tradução automática, sumarização de documentos e resposta a perguntas. A capacidade de capturar relações de longo alcance nos textos representa um dos diferenciais desses modelos em relação às arquiteturas anteriores.

A introdução dos Transformers, realizada por Vaswani et al. [166] no artigo seminal “Attention is All You Need”, representou uma mudança fundamental no campo do Processamento de Linguagem Natural. A principal inovação foi o mecanismo de atenção, que atribui pesos diferentes às palavras de entrada de acordo com seu contexto, permitindo identificar quais termos são mais relevantes em cada situação. Essa característica torna possível o processamento paralelo de sequências longas, o que acelera substancialmente o treinamento e aumenta a eficácia na modelagem de dependências distantes.

Antes dessa arquitetura, os modelos de PLN eram dominados por RNNs (Recurrent Neural Networks) e LSTMs (Long Short-Term Memory Networks). Embora funcionais, esses métodos apresentavam limitações significativas: dificuldade em capturar dependências de longo alcance devido ao problema do vanishing gradient e restrições de desempenho, já que processavam o texto de maneira estritamente sequencial, reduzindo o potencial de paralelização. O *Transformer* superou essas limitações ao considerar todas as palavras de entrada simultaneamente, ponderando sua importância relativa por meio da atenção.

Com isso, os LLMs não apenas aumentaram a precisão em tarefas tradicionais

de PLN, como também ampliaram o escopo de aplicações, abrangendo geração criativa de texto, tradução contextualizada e correção gramatical automática. Por outro lado, o avanço desses modelos também trouxe questionamentos relacionados à ética, viés e uso responsável da inteligência artificial, aspectos que vêm sendo discutidos pela comunidade científica.

Arquitetura Geral do Transformer

Os LLMs modernos fundamentados na arquitetura *Transformer* são a base de inúmeros avanços no processamento de linguagem natural porque oferecem uma forma eficiente e escalável de lidar com sequências de texto. Diferente de abordagens anteriores, que processavam as palavras uma a uma em ordem, os *Transformers* permitem que todo o contexto seja considerado em paralelo, o que torna o treinamento e a inferência mais rápidos e flexíveis [166].

A estrutura geral de um *Transformer* é composta por blocos que se repetem, cada um deles combinando mecanismos de atenção e redes neurais totalmente conectadas (feedforward). Em sua formulação original, a arquitetura possui dois grandes componentes: o encoder, responsável por transformar a sequência de entrada em uma representação interna, e o decoder, que utiliza essa representação para gerar uma saída, como no caso da tradução automática. Embora alguns modelos atuais utilizem apenas a parte do encoder (como o BERT) ou apenas o decoder (como o GPT), o princípio fundamental continua o mesmo: ambos dependem intensamente do mecanismo de atenção. Cada camada do encoder e do decoder contém três elementos centrais:

- Mecanismo de atenção múltipla (Multi-Head Attention): avalia, em paralelo, diferentes formas de relacionar as palavras entre si.
- Redes feedforward: transformam as representações intermediárias, permitindo maior capacidade de modelagem.
- Normalização e conexões residuais: estabilizam o treinamento e ajudam a preservar informações ao longo das camadas.

O conceito de *self-attention* é um dos pontos mais inovadores. Nesse mecanismo, cada palavra da sequência não é processada isoladamente, mas em com-

paração com todas as outras palavras da mesma sequência. Isso permite que o modelo entenda relações de dependência de longo alcance, como entre o sujeito no início de uma frase e o verbo que aparece muito depois. O *multi-head attention* amplia essa ideia ao aplicar várias atenções em paralelo, cada uma aprendendo a capturar um tipo de relação — algumas cabeças podem identificar proximidade sintática, enquanto outras captam conexões semânticas mais distantes.

Um aspecto central da arquitetura é que todo esse processamento ocorre de forma paralela. Diferente das redes recorrentes (RNNs) ou das LSTMs, que analisavam o texto palavra por palavra em sequência, o *Transformer* avalia a frase inteira de uma só vez. Essa mudança de paradigma permitiu que modelos fossem treinados em coleções massivas de texto, chegando a bilhões de parâmetros e escalando para tamanhos sem precedentes.

Além de sua eficiência estrutural, o *Transformer* também é flexível: pode ser adaptado a diferentes tarefas com pequenas modificações. Modelos autoregressivos, como a família GPT, são baseados apenas no decoder e são treinados para prever a próxima palavra em uma sequência de forma unidirecional. Já os modelos bidirecionais, como o BERT, exploram tanto o contexto à esquerda quanto o à direita, o que os torna particularmente eficazes em tarefas de compreensão textual.

O impacto da arquitetura *Transformer* é amplo e vai além do processamento de linguagem natural em sua forma mais básica. A combinação entre paralelismo, capacidade de capturar dependências de longo alcance e flexibilidade estrutural abriu caminho para a criação de famílias inteiras de modelos com diferentes finalidades. Alguns deles utilizam apenas o encoder, outros apenas o decoder, e há ainda os que combinam os dois de formas distintas. Essa diversidade deu origem a arquiteturas conhecidas, como BERT, GPT, T5 e LLaMA, que se tornaram referências na área e serão discutidas a seguir em maior detalhe.

BERT (Bidirectional Encoder Representations from Transformers)

O BERT (Bidirectional Encoder Representations from Transformers) introduziu uma nova forma de treinamento em larga escala, na qual o modelo considera tanto o contexto à esquerda quanto o contexto à direita de uma palavra-alvo. Essa abor-

dagem permitiu uma compreensão mais rica da linguagem em comparação com modelos anteriores, que eram predominantemente unidirecionais.

O BERT é construído sobre a pilha de encoders do Transformer. Isso significa que ele não é projetado para gerar texto de forma autoregressiva, mas sim para produzir representações contextuais profundas de palavras e frases, que podem depois ser utilizadas em diversas tarefas de PLN. Essa característica torna o BERT especialmente adequado para problemas de compreensão textual, como classificação, resposta a perguntas e extração de informações.

O modelo é pré-treinado em uma grande quantidade de texto utilizando duas tarefas principais:

- Masked Language Modeling (MLM): Palavras aleatórias em uma sequência são substituídas por um token especial de máscara, e o modelo é treinado para prever as palavras originais com base no restante do contexto. Esse procedimento permite que o BERT aprenda relações bidirecionais, uma vez que a previsão de uma palavra depende de termos anteriores e posteriores na frase.
- Next Sentence Prediction (NSP): O modelo recebe pares de frases e deve identificar se a segunda frase segue logicamente a primeira. Essa tarefa ajuda o BERT a capturar relações discursivas entre sentenças, algo essencial em aplicações como resposta a perguntas e inferência textual.

Após o pré-treinamento, o BERT pode ser ajustado “finamente” (Fine-Tuning) para tarefas específicas, muitas vezes com datasets muito menores que aqueles usados no pré-treinamento. Esse processo de adaptação é o que permite que um único modelo, previamente treinado de maneira genérica, seja aplicado em uma ampla gama de problemas.

O BERT rapidamente estabeleceu novos patamares em benchmarks de PLN, como GLUE (General Language Understanding Evaluation) e SQuAD (Stanford Question Answering Dataset). Sendo o SQuAD um dos conjuntos de dados mais utilizados na tarefa de PLN de geração de perguntas e respostas, segundo revisão [28]. Sua eficácia impulsionou o desenvolvimento de uma série de variações e aprimoramentos, como RoBERTa, DistilBERT e ALBERT, que buscaram melhorar

desempenho, eficiência ou reduzir custos de treinamento.

O BERT é amplamente utilizado em:

- Classificação de Texto: Análise de sentimento, detecção de spam, categorização de documentos.
- Respostas a Perguntas: Modelos que identificam trechos relevantes em um texto para responder perguntas formuladas em linguagem natural.
- Extração de Informações: Identificação de entidades nomeadas (pessoas, lugares, organizações) e relações entre elas.

Apesar de seu impacto, o BERT apresenta limitações relevantes. O modelo é pesado em termos computacionais, exigindo grande capacidade de processamento para treinamento e mesmo para inferência em aplicações práticas. Além disso, sua janela de contexto é limitada, o que dificulta o processamento de documentos muito extensos sem estratégias adicionais de segmentação.

GPT (Generative Pre-trained Transformer)

O Generative Pre-trained *Transformer* (GPT) é um dos LLMs mais conhecidos. Ele é treinado de forma autoregressiva, o que significa que prediz a próxima palavra em uma sequência, dada a entrada anterior. Isso o torna excelente para tarefas de geração de texto. O GPT 3.5 foi uma versão inicial do GPT, provavelmente era uma versão ajustada do GPT-3 de 175B de parâmetros. Ele mostrou que, ao ser treinado em grandes quantidades de texto, poderia gerar conteúdo parecido com o texto gerado por humanos. Esse modelo pode realizar uma ampla gama de tarefas de PLN sem a necessidade de ajustes finos específicos, simplesmente recebendo exemplos de como a tarefa deve ser executada (aprendizado por poucos exemplos, ou few-shot learning).

O GPT tem sido aplicado em diversas áreas, incluindo:

- Geração de Texto: Criação de conteúdo, histórias, artigos, etc.
- Assistentes Virtuais: Implementação de sistemas de diálogo baseados em IA.
- Tradução Automática: Utilização de contexto amplo para melhorar a tradução entre idiomas.

O GPT é um modelo autoregressivo que se concentra na geração de texto. É

treinado para prever a próxima palavra em uma sequência, o que o torna excelente para tarefas de geração de texto, como chatbots.

Além de BERT e GPT, há muitos outros modelos baseados em *Transformers* projetados para tarefas específicas. Alguns exemplos incluem o T5 (Text-To-Text Transfer Transformer) que converte qualquer tarefa de PLN em um problema de tradução; o XLNet que combina ideias de BERT e *Transformers* autoregressivos para melhorar a modelagem de dependências de longo alcance; o RoBERTa (A Robustly Optimized BERT Pretraining Approach) que é uma variação do BERT com treinamento aprimorado, além do DistilBERT. O T5 transforma qualquer tarefa de PLN em um problema de tradução, onde a entrada e a saída são tratadas como texto. Isso simplifica o *Fine-Tuning* para diferentes tarefas.

Cabe ressaltar ainda o XLNet e o DistilBERT. O XLNet combina vantagens dos modelos autoregressivos e bidirecionais, como o GPT e BERT, para capturar dependências de longo alcance de forma mais eficiente. Já o DistilBERT é uma versão reduzida do BERT, com menos parâmetros, mas mantendo uma alta performance, o que o torna mais eficiente para uso em produção.

distilbert-base-uncased

O modelo distilbert-base-uncased [3] foi lançado em 2019, sendo menor e mais rápido que o BERT e otimizado para tarefas que processam frases ou sentenças. Sua versão uncased é útil para descrições de código e problemas, já que não considera diferenças de maiúsculas/minúsculas. Ele é pré-treinado em um grande corpus, o que ajuda na generalização, e tem uma boa arquitetura para esse tipo de tarefa, além de exigir hardware acessível.

O distilbert resulta de um processo de knowledge distillation que reduz em aproximadamente 40% o número de parâmetros e acelera a inferência em cerca de 60%, preservando 95–97% da acurácia do BERT-base (original) em benchmarks de compreensão de linguagem natural [125]. Essa compacidade de fornecer representações ricas com custo computacional inferior é particularmente vantajosa em ambientes de hardware moderado. Com apenas 67M de parâmetros [125], a variante uncased cabe em uma GPU modesta (por exemplo, 6 GB de RAM da placa

de vídeo), possibilitando ajuste fino e inferência dentro de recursos computacionais restritos. Assim, torna-se viável re-treinar o modelo à medida que novos dados de projeto se acumulam, mantendo a acurácia sem investir em infraestruturas onerosas.

Por fim, o amplo suporte no ecossistema Hugging Face para os modelos do tipo BERT e para outros modelos simplifica a reprodutibilidade e a integração em pipelines de tarefas de processamento de linguagem natural.

Fine-Tuning de Modelos Pré-Treinados

Modelos pré-treinados como BERT e GPT demonstraram capacidade de compreender nuances semânticas em texto e transferir esse conhecimento para diversas tarefas específicas através de *Fine-Tuning* [85].

O *Fine-Tuning* de modelos pré-treinados é uma técnica fundamental no Processamento de Linguagem Natural (PLN) moderno, especialmente ao trabalhar LLMs. *Fine-Tuning* permite adaptar um modelo geral para tarefas específicas, como classificação de texto, análise de sentimentos ou geração de linguagem, utilizando um conjunto de dados menor e específico.

Fine-Tuning é o processo de tomar um modelo pré-treinado em uma grande quantidade de dados gerais e adaptá-lo para uma tarefa específica. Este processo envolve ajustar os pesos do modelo, mas com uma taxa de aprendizado menor para não “desaprender” o que foi aprendido durante o pré-treinamento. Modelos como BERT [35] 2019, GPT [115], e T5 [116] são comumente fine-tuned para tarefas específicas.

Uma alternativa eficiente ao *Fine-Tuning* completo é o uso de métodos de adaptação com baixo número de parâmetros, como o LoRA (Low-Rank Adaptation) [57]. Em vez de atualizar todos os pesos do modelo pré-treinado, o LoRA introduz pequenas matrizes adicionais de baixa dimensão que são ajustadas durante o treinamento, enquanto os pesos originais permanecem congelados. Essa técnica reduz drasticamente o número de parâmetros que precisam ser treinados, diminuindo o custo computacional e de armazenamento. Na prática, isso torna viável aplicar *Fine-Tuning* em LLMs muito grandes, mesmo em ambientes com recursos limita-

dos, preservando boa parte do desempenho obtido pelo ajuste completo.

A principal vantagem do *Fine-Tuning* é a eficiência, pois permite que os modelos aprendam rapidamente uma nova tarefa, utilizando relativamente poucos dados. Além disso, modelos pré-treinados já capturam padrões linguísticos gerais, o que torna o *Fine-Tuning* uma abordagem útil para resolver problemas específicos sem precisar treinar um modelo do zero. O processo de *Fine-Tuning* geralmente envolve os seguintes passos:

- Escolha do Modelo: Selecionar um modelo pré-treinado adequado para a tarefa. Modelos como BERT e GPT são populares devido à sua versatilidade.
- Preparação dos Dados: Os dados precisam estar formatados de maneira que sejam compatíveis com a tarefa específica, como classificação de texto ou resposta a perguntas.
- Configuração do Treinamento: Ajuste de hiperparâmetros como a taxa de aprendizado, número de épocas e tamanho do lote.
- Treinamento: Executar o treinamento do modelo no conjunto de dados específico.
- Avaliação: Avaliar o desempenho do modelo ajustado em um conjunto de validação ou teste.

Cabe ressaltar que a biblioteca *Transformers* da Hugging Face tornou-se a ferramenta de referência para trabalhar com modelos baseados em Transformers. Ela oferece uma ampla gama de modelos pré-treinados que podem ser facilmente integrados em pipelines de PLN. Embora o *Fine-Tuning* seja uma técnica com potencial, é importante considerar alguns desafios:

- Overfitting: Ajustar demais o modelo para os dados de treinamento específicos pode reduzir a generalização para novos dados.
- Biases Inerentes: Se o modelo pré-treinado já contém vieses, o *Fine-Tuning* pode reforçá-los, especialmente se os dados de treinamento forem limitados ou enviesados.
- Requisitos Computacionais: *Fine-Tuning* de LLMs pode ser computacionalmente intensivo, especialmente para modelos maiores como GPT-3.
- Além disso, algumas abordagens avançadas para melhorar o processo de

Fine-Tuning incluem:

  - Learning Rate Warmup: Aumentar gradualmente a taxa de aprendizado no início do treinamento para evitar grandes atualizações de peso que poderiam desestabilizar o modelo.
  - Layer-Wise Learning Rate Decay: Aplicar diferentes taxas de aprendizado para diferentes camadas do modelo, com camadas inferiores aprendendo mais lentamente.
  - Data Augmentation: Aumentar a diversidade do conjunto de dados de treinamento para melhorar a robustez do modelo.

- O *Fine-Tuning* tem uma vasta gama de aplicações em PLN, incluindo:
  - Classificação de Texto: Análise de sentimentos e categorização de notícias.
  - Respostas a Perguntas: Modelos que respondem a perguntas baseadas em um contexto textual específico.
  - Geração de Texto: Para geração de textos específicos de um domínio, como redação de artigos científicos.
  - Tradução Automática: Adaptação de modelos de tradução para dialetos ou linguagens específicas.

Abordagens para estimativa, Zero Shot e Few Shot

Diante destes avanços, pesquisadores começaram a investigar se LLMs poderiam melhorar a predição de *Story Points* a partir da descrição textual das *User Stories*, em comparação com métodos tradicionais de estimativa de esforço [160, 4].

Antes do advento dos LLMs e do aprendizado profundo no contexto de estimativas ágeis, diversas abordagens tradicionais de aprendizado de máquina foram exploradas para prever *Story Points* a partir de dados históricos de projetos [23, 111, 130]. Essas abordagens costumam tratar a estimativa como um problema de regressão supervisionada (quando os *Story Points* são valores contínuos ou ordinais) ou de classificação em faixas (ou discretizações) de esforço.

Já foi estudado, para o problema de estimativa de Story Points, o uso de vetores de características de texto baseados em TF-IDF das descrições de issues

do Jira [9], combinados com um classificador SVM para prever o número de Story Points [111]. Naquele trabalho, os autores relataram, em 2016, resultados promissores, superando estimativas *baseline* simples ao usar a média ou a mediana. Essa técnica serviu de referência inicial, contra a qual trabalhos subsequentes com aprendizado profundo seriam comparados [172].

Outro estudo explorou algoritmos de regressão, ensemble e máquinas de vetores de suporte [137]. Por exemplo, alguns pesquisadores aplicaram máquinas de vetores de suporte usando atributos textuais e atributos de legibilidade [96]. Um outro modelo investigou até que ponto as características dos desenvolvedores poderiam explicar a variância nos Story Points [130]. De modo geral, esses métodos mais tradicionais obtiveram desempenho limitado, muitas vezes com erros absolutos médios altos, indicando dificuldade em capturar a complexidade semântica das *User Stories* de usuário apenas com atributos gramaticais, tais como a contagem de palavras [46].

Zero Shot: A abordagem LLM zero-shot learning refere-se à capacidade dos LLMs de resolver tarefas sem a necessidade de exemplos explícitos fornecidos durante a inferência. Nesse contexto, a tarefa de processamento de linguagem natural (text-classification) é especificada unicamente por meio de uma instrução textual (prompt), e o modelo deve inferir a ação esperada com base em seu conhecimento prévio adquirido durante o pré-treinamento [115].

Few Shot: Já o few-shot learning caracteriza-se pela inclusão de um pequeno conjunto de exemplos da tarefa no próprio prompt, com o objetivo de guiar a geração do modelo durante a inferência. Essa técnica permite ao modelo identificar padrões desejados com base nos exemplos fornecidos e aplicá-los a novos casos, mesmo sem reconfiguração ou ajuste de parâmetros. Trata-se de uma abordagem intermediária entre o zero-shot e o treinamento supervisionado tradicional, sendo especialmente eficaz em tarefas de classificação com variações contextuais [118]. Sua principal vantagem está na adaptação rápida a novas tarefas, com custo computacional reduzido.

Brown et al. [17] introduziu o GPT-3, mostrando o poder dos grandes modelos de linguagem (LLMs) e do few-shot learning, que são extensões práticas de transfer

learning em larga escala.

Ajuste Fino com Transfer Learning

O *Transfer Learning* (ou aprendizado por transferência) é uma técnica de aprendizagem de máquina que visa reaproveitar o conhecimento adquirido por um modelo pré-treinado em uma tarefa de origem para resolver uma nova tarefa, geralmente relacionada, denominada tarefa-alvo. Essa abordagem parte do princípio de que as representações aprendidas em grandes bases de dados podem ser úteis em diferentes contextos, reduzindo o custo computacional e a necessidade de dados rotulados [105].

Em vez de treinar um modelo do zero, o transfer learning utiliza um modelo base previamente treinado (por exemplo, *BERT* ou *GPT*) e ajusta seus parâmetros por meio de uma etapa chamada *fine-tuning*. Nessa etapa, o modelo é refinado com um conjunto de dados específico da tarefa-alvo, permitindo que as representações previamente aprendidas sejam especializadas para o novo domínio.

De modo geral, o processo de aprendizado por transferência envolve as seguintes etapas principais:

- Seleção do modelo base: escolha de um modelo pré-treinado em uma base ampla e genérica, que contenha características relevantes para a nova tarefa.
- Reutilização ou congelamento de camadas: as camadas iniciais do modelo, responsáveis por aprender representações genéricas, podem ser mantidas fixas, enquanto as camadas finais são ajustadas.
- Ajuste fino (*fine-tuning*): o modelo é reentreinado com um conjunto de dados menor e específico, adaptando suas representações às peculiaridades da nova tarefa.

A principal vantagem do *transfer learning* reside na capacidade de reduzir significativamente o tempo de treinamento e o volume de dados necessários, além de melhorar o desempenho em contextos onde os dados são escassos. No entanto, sua eficácia depende da similaridade entre as tarefas de origem e de destino; quando há grande discrepância entre os domínios, pode ocorrer o fenômeno de *negative transfer*, no qual o conhecimento prévio prejudica o aprendizado.

Em aplicações de Processamento de Linguagem Natural (PLN), o *transfer learning* tem se mostrado especialmente eficaz. Modelos como o *BERT* [35] e o *GPT* [17] são treinados em grandes corpora textuais e, posteriormente, ajustados para tarefas específicas, como, por exemplo, estimar esforço. Essa estratégia permite que modelos de linguagem alcancem alto desempenho mesmo em domínios especializados com poucos exemplos rotulados.

2.7 Aplicações dos Fundamentos nesta Tese

Neste capítulo, foram apresentados conceitos que sustentam o desenvolvimento das soluções nos capítulos seguintes. Por exemplo, a seção sobre “User Stories” fornece a base conceitual para entender como os requisitos são descritos e gerenciados em contextos ágeis. Isso é fundamental para a coleta e estruturação dos dados no NeoDataset (Capítulo 5). Esses conceitos são aplicados diretamente na caracterização e no processamento das *User Stories*, servindo como insumos tanto para os modelos preditivos quanto para a aplicação de recomendações de melhorias textuais (Capítulo 6, Capítulo 7 e Capítulo 8). A compreensão de como as *User Stories* são formuladas e utilizadas no ciclo de desenvolvimento representa o ponto de partida para todos os experimentos realizados.

Os “índices de legibilidade” e as métricas linguísticas discutidas neste capítulo foram aplicados no “Neo Legibility Effort Model” (Capítulo 7). Nesta aplicação, os índices como Flesch Reading Ease, Gunning Fog e Dale–Chall são utilizados como variáveis preditoras para estimar o esforço em Story Points. Esses mesmos conceitos também embasam a parte do “Neo *User Story* Tutor” (Capítulo 6), que avalia e sugere melhorias nas descrições, buscando produzir textos mais claros e objetivos.

Os conceitos sobre PLN e LLM, apresentados no final deste capítulo, foram aplicados no “Neo LLM Predictor” (Capítulo 8) e no módulo de recomendação do “Neo *User Story* Tutor” (Capítulo 6). Técnicas como tokenização, embedding e estratégias de uso de LLMs (zero-shot, few-shot e fine-tuning) são exploradas para estimar *Story Points* e gerar sugestões de reescrita que sejam mais adequadas. A teoria sobre vetorização de texto, incluindo TF-IDF e embeddings contextuais, foi

utilizada em diversas escolhas metodológicas para a comparação entre abordagens tradicionais e aquelas baseadas em LLM.

Capítulo 3

Trabalhos Relacionados

Begin! Be bold and venture to be wise.

Quintus Horatius Flaccus

3.1 Introdução

Este capítulo sintetiza o estado da arte em estimativa de esforço no desenvolvimento de software, com foco em abordagens baseadas em Aprendizagem de Máquina (AM) e Processamento de Linguagem Natural (PLN) aplicadas à previsão de *Story Points* a partir de *User Stories*. Apresentam-se modelos, artefatos e métricas empregados, discutindo criticamente os avanços, limitações e lacunas. Este capítulo estende a revisão sistemática disponível no Apêndice C.

3.2 Estado da Arte e Problemas

Os modelos clássicos de estimativa de esforço são baseados em modelos algorítmicos (e.g., COCOMO, Pontos de Função e Use Case Points) e têm sido amplamente utilizados para estimativas de esforço com variáveis estruturadas (Apêndice C). Em ambientes ágeis, a adoção de *User Stories* e *Story Points* introduziu uma forma de verificação de esforço mais simples; porém, manteve a dependência do julgamento humano, com consequências de subjetividade e inconsistência entre

times e projetos.

Com a popularização das *User Stories* como artefatos textuais centrais, surgiram preditores baseados em texto. O TFIDF-SE [111] constitui uma *baseline* amplamente reproduzida: o texto é vetorizado por TF-IDF e um regressor (tipicamente SVR) que estima os SP. Em seguida, DEEP-SE [23] empregou redes recorrentes (e.g., LSTM) para capturar dependências semânticas de sequência, reportando ganhos em relação ao TFIDF-SE. Contudo, reproduções posteriores [145] contestaram essa superioridade e reafirmaram o TFIDF-SE como referência, expondo a instabilidade da conclusão entre estudos (sensibilidade a variações de dados, pré-processamento e protocolo experimental).

Outros estudos no mesmo domínio (estimativa de esforço em *Story Points* a partir do texto da user Story) já foram apresentados por outros pesquisadores [81, 70]. Geralmente, nestes estudos, a comparação com trabalhos relacionados é realizada reproduzindo os experimentos selecionados como baseline, incluindo um ou vários modelos e o novo modelo sugerido. Sendo que esta comparação pode ser realizada tanto no mesmo conjunto de dados utilizado no *baseline* quanto em um novo conjunto de dados.

Por exemplo, Marapelli et al. [81] apresentam um modelo de aprendizado profundo com RNN-CNN (Rede Neural Recorrente - Rede Neural Convolucional) para a estimativa de *Story Points* no desenvolvimento ágil de software. Os autores usaram o conjunto de dados de Choetkiertikul et al. [23] (nesta Tese chamado de ChoetkiertikulDataset) e propõem o uso de uma Long Short-Term Memory Bidirecional (BiLSTM), que processa a descrição da *User Story* em direções para frente e para trás, combinada com uma CNN para a extração precisa de características do texto. O objetivo do estudo de Marapelli et al. [81] é aprimorar a precisão na predição do esforço necessário para completar uma tarefa, medida em Story Points, utilizando o texto descritivo da *User Story* como entrada. Os resultados experimentais, comparando-se com modelos existentes, como o LD-RNN de Choetkiertikul et al. [23], indicam que o modelo RNN-CNN proposto oferece um desempenho superior aos anterioremente aqui discutidos na estimativa de *Story Points* em vários conjuntos de dados de projetos. Em essência, o trabalho de Marapelli et al. [81] busca

aplicar técnicas avançadas de processamento de linguagem natural e aprendizado profundo para resolver o desafio da estimativa de esforço em ambientes de desenvolvimento ágil.

Já Cheemaa et al. [20] apresenta o Efficient GPT for *Story Point* Estimation (EGPT-SPE), um novo algoritmo para automatizar a estimativa de Story Points, visando maior precisão e eficiência. O estudo aborda a dificuldade e as consequências negativas das estimativas imprecisas de esforço em projetos de software, incluindo estouros de orçamento e atrasos. Para melhorar o processo, o EGPT-SPE otimiza o modelo pré-treinado GPT-2, removendo cabeças de atenção ineficientes, o que resulta na redução de custos computacionais e no aumento da acurácia. A eficácia do algoritmo é demonstrada por meio de experimentos em dois conjuntos de dados extensos, o Choetkiertikul e o TAWOS, nos quais o EGPT-SPE superou métodos de ponta, incluindo seu predecessor, o GPT2SP [46], em avaliações tanto *within-project* quanto *Cross-Project*.

O GPT2SP de Fu and Tantithamthavorn [46] é uma abordagem baseada em *Transformer* e no modelo GPT2 pré-treinado, utilizada para a estimativa de Story Points. Os autores destacam que os métodos de estimativa existentes, como o Deep-SE, são imprecisos, subjetivos e não transferíveis a outros projetos. Para superar essas limitações, propõem o GPT2SP, que oferece maior precisão e interpretabilidade. A avaliação demonstra que o GPT2SP é significativamente mais preciso do que as dez *baselines* existentes, incluindo o Deep-SE. Além disso, um estudo de pesquisa realizado com 16 profissionais Agile sugere que o recurso de Explicações por IA (XAI) do GPT2SP é percebido como mais útil e confiável, destacando a necessidade prática de ferramentas explicáveis na engenharia de software.

O relacionamento dos trabalhos nesta seção será explorado na próxima e na seção 3.4, por meio da análise de lacunas nas pesquisas neles realizadas.

3.3 Lacunas

As lacunas e desafios encontrados nos métodos existentes para a estimativa de *Story Points* são diversos, abrangendo desde a subjetividade humana até os cus-

tos computacionais dos modelos [20]. Existem lacunas em precisão, validação e suporte à escala nos métodos de estimativa de esforço em desenvolvimento ágil, como a validação empírica insuficiente, a precisão limitada de modelos automatizados e a inadequação para contextos organizacionais complexos. Algumas dessas lacunas identificadas são apresentadas à seguir:

1. Inconsistência e Imprecisão nas Estimativas. O contexto geral é que a avaliação imprecisa dos *Story Points* representa um problema a ser solucionado, pois pode levar a:

- A perda de prazos para marcos intermediários pode resultar em compromissos significativos no cronograma do projeto.
- Atrasos gerais na conclusão do projeto.
- Perdas tanto em despesas quanto em reputação.
- Grandes projetos de software excedem seus orçamentos.

2. Limitações das Técnicas Manuais e Baseadas em Especialistas. As abordagens tradicionais, que se baseiam no conhecimento humano ou em dados históricos, apresentam várias deficiências:

- Aplicabilidade Limitada: Modelos de estimativa desenvolvidos com base em dados de vários projetos passados têm uma aplicabilidade limitada a projetos que compartilham características semelhantes.
- Subjetividade e Erro Humano: A estimativa manual dos *Story Points* impacta significativamente a precisão e a eficiência do processo devido a fatores como erro humano, inexperiência e divergências entre a equipe de estimativa.
- Custo e Tempo: A construção de avaliações baseadas na opinião de especialistas é demorada e cara, e o sucesso exige acesso constante aos profissionais necessários.
- Inconsistência com o Esforço Real: Alguns estudos encontram uma baixa correlação entre os *Story Points* estimados por especialistas humanos e o tempo real de desenvolvimento. Isso sugere que as estimativas de *Story Points* podem ser inconsistentes e refletir apenas parcialmente a quantidade real de trabalho envolvido no desenvolvimento.

3. Falhas nos Métodos de Aprendizagem de máquina e aprendizado profundo.

As abordagens automatizadas, apesar de seu potencial, também enfrentam problemas que limitam sua eficácia no mundo real:

- Métodos Convencionais de ML: Frequentemente, falham em prever com precisão devido à falta de compreensão contextual dos requisitos do usuário.
- Aprendizado Profundo (ou Deep Learning - DL): Enfrenta desafios relacionados à alta complexidade temporal e custos computacionais significativos ao extrair padrões do texto de linguagem natural dos requisitos do usuário.
- Complexidade Injustificada: Pesquisas indicaram que alguns métodos avançados que utilizam técnicas como Latent Dirichlet Allocation (LDA) e clustering têm desempenho semelhante aos métodos de ponta, mas nenhuma das técnicas supera significativamente os estimadores mais simples, levantando dúvidas sobre a complexidade das abordagens mais avançadas.

A revisão conduzida por Uc-Cetina [160] evidencia avanços significativos na utilização de técnicas de aprendizado de máquina para a estimativa de esforço em projetos de software. Contudo, observa-se um conjunto de lacunas e desafios persistentes que ainda limitam a maturidade e a aplicabilidade prática desses modelos.

Em primeiro lugar, destaca-se a baixa capacidade de generalização dos modelos atuais, que apresentam desempenho insatisfatório quando aplicados a projetos distintos dos utilizados no treinamento. Tal limitação reforça a necessidade de investigar abordagens mais robustas de transfer learning e adaptação de domínio, capazes de lidar com contextos heterogêneos de desenvolvimento.

Outra lacuna refere-se à predição baseada exclusivamente em dados textuais, como descrições de *User Stories*. Estudos recentes indicam que a análise semântica isolada não é suficiente para capturar a complexidade da estimativa de esforço, sendo necessário combinar características textuais e estruturais, tais como métricas de código, histórico da equipe e contexto organizacional.

Além disso, observa-se a escassez de modelos híbridos que integrem métodos estatísticos, redes neurais e algoritmos evolucionários de maneira sinérgica. A literatura ainda carece de abordagens que incorporem fatores contextuais e organizacionais — como maturidade de processos e papéis das equipes — para melhorar a acurácia e a interpretabilidade dos resultados.

No contexto da metodologia ágil, há também uma carência de investigações voltadas à estimativa de esforço em nível individual, considerando os padrões de produtividade e erro de cada engenheiro. Essa dimensão microanalítica poderia favorecer modelos personalizados e mecanismos de feedback automatizados para o aprimoramento contínuo das estimativas.

Outro ponto crítico é a limitação dos conjuntos de dados disponíveis. Apenas dois grandes conjuntos — ChoetkiertikulDataset e TAWOSDataset — são amplamente utilizados na literatura, o que restringe a capacidade de comparação entre estudos e compromete a reprodutibilidade dos resultados. Assim, torna-se urgente o desenvolvimento de novos repositórios públicos, diversificados e bem anotados.

Adicionalmente, o estudo ressalta o potencial inexplorado das arquiteturas baseadas em *Transformers* (como BERT e GPT), amplamente bem-sucedidas em tarefas de Processamento de Linguagem Natural. A aplicação dessas arquiteturas à estimativa de esforço constitui uma oportunidade promissora para representar contextos complexos e dependências semânticas entre *User Stories*.

No âmbito metodológico, identificam-se também lacunas relacionadas à explicabilidade e interpretabilidade dos modelos. Poucos trabalhos abordam a compreensão das razões subjacentes às previsões, aspecto essencial para a aceitação gerencial das estimativas produzidas por algoritmos de aprendizado de máquina.

A ausência de métricas e protocolos de avaliação padronizados constitui outra limitação importante. A diversidade de métricas utilizadas — como MAE, MRE e RMSE — e a falta de consenso sobre estratégias de validação dificultam comparações diretas entre estudos.

Do ponto de vista prático, nota-se a falta de integração dos modelos de predição em ferramentas de gestão reais, como Jira ou GitLab. São necessários estudos empíricos que avaliem o impacto dessas integrações na tomada de decisão e na produtividade das equipes.

Por fim, há carência de investigações longitudinais voltadas à evolução temporal das estimativas. Pouco se sabe sobre como a experiência acumulada de engenheiros e equipes influencia a redução progressiva de erros de predição ao longo do tempo.

Em síntese, as lacunas identificadas por Uc-Cetina [161] abrem múltiplas frentes de pesquisa, destacando a necessidade de modelos mais generalizáveis, interpretáveis e integrados ao contexto real de desenvolvimento ágil, além de bases de dados mais amplas e diversificadas para sustentar tais avanços.

Outra lacuna é o uso de Atributos linguísticos e legibilidade. Além de representações distribuídas (e.g., *embeddings*), atributos linguísticos explícitos (legibilidade, subjetividade e sentimento) são subexplorados na engenharia de software. Há evidências em domínios correlatos de que tais atributos capturam a carga cognitiva e a clareza textual, potencialmente relacionadas ao esforço [24]. Trabalhos que abordaram esse aspecto o fizeram de modo limitado (por exemplo, uso pontual do Gunning Fog [22]), deixando margem para investigações sistemáticas e comparativas.

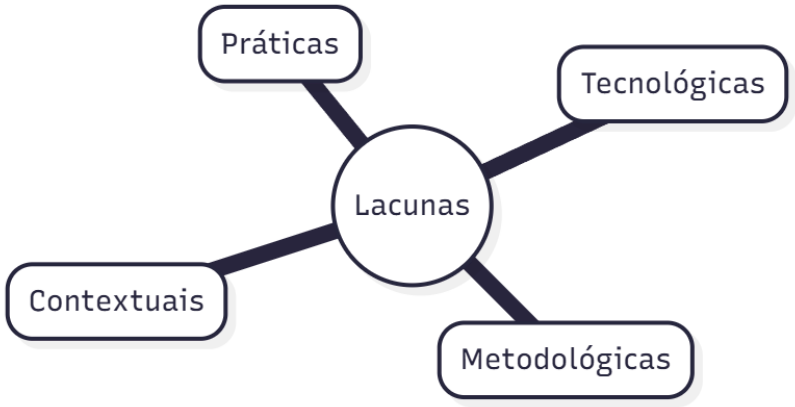
Além disso, a aplicação de LLMs para estimativa e suporte à escrita também é uma lacuna. Modelos de Linguagem de Grande Escala (LLMs) habilitam duas frentes: (i) predição direta de *Story Points* (via zero-shot, few-shot e fine-tuning) e (ii) tutoria de texto para melhorar a qualidade das *User Stories* antes da estimativa. Avanços recentes incluem o EGPT-SPE [20], que otimiza cabeças de atenção e relata ganhos de acurácia com menor custo computacional. Mesmo assim, ainda há carência de estudos que comparem, de forma controlada, LLMs ajustados versus *baselines* clássicos sob múltiplos conjuntos de dados e métricas homogêneas.

Por fim, ainda existem lacunas relacionadas aos conjuntos de dados e à reprodutibilidade. A comparabilidade entre estudos é frequentemente limitada por conjuntos de dados proprietários e protocolos heterogêneos. Assim, um conjunto de dados públicos, extraído de repositórios de código aberto como GitLab/Jira, com *User Stories* e rótulos de Story Point, é essencial para reduzir viés, favorecer o reuso e permitir uma avaliação robusta de modelos (tradicionais, profundos e LLMs).

Como são muitas, uma possível divisão das lacunas poderia ser feita ao longo das seguintes classes: Metodológicas, Práticas, Tecnológicas e Contextuais (veja Figura 3.1).

- Lacunas Metodológicas
  - Dependência Excessiva de Técnicas Subjetivas.
    - Problema: Predominância de métodos baseados em julgamento hu-

Figura 3.1: Categoria das Lacunas



mano (planning poker, estimativas por analogia, avaliação do especialista).

  * Impacto: Amplifica vieses cognitivos e inconsistências na mensuração do esforço.
  * Evidência: Estudos mostram que usuários relatam erros médios consideraveis nas estimativas [43].

– Métricas de Acurácia Inadequadas
  * Problema: Uso extensivo de métricas tradicionais (MMRE/MRE) que não resolvem o problema da baixa precisão.
  * Impacto: Dificuldade em comparar a efetividade de diferentes abordagens.
  * Evidência: Revisões sistemáticas mostram que a precisão geral e erro médio absoluto continua sendo uma métrica de comparação válida [107].

– Validação e Generalização Fracas
  * Problema: Modelos avaliados apenas *within-project*, com pouca replicação entre projetos.
  * Impacto: Limitação severa da aplicabilidade prática.
  * Evidência: Estudos de replicação mostram que modelos como DeepSE raramente generalizam bem entre projetos [70].

– Ausência de Consenso sobre Drivers e Métricas.
  * Problema: Falta de acordo sobre quais cost drivers e métricas de

tamanho usar consistentemente.
    - Impacto: Impede comparações e reprodutibilidade entre estudos.
    - Evidência: Literatura mostra inconsistência na definição e uso de fatores de custo [32].

- Lacunas Práticas.
  - Viés Humano e Erros Sistemáticos.
    - Problema: Sub/superestimações frequentes por fatores psicológicos e organizacionais.
    - Impacto: Redução da confiabilidade das estimativas na prática.
  - Alto Custo de Tempo das Técnicas.
    - Problema: Métodos colaborativos, como o Planning Poker, consomem tempo excessivo.
    - Impacto: Redução da produtividade das equipes.
    - Evidência: Outros métodos também podem ser eficientes [128].
  - Suporte Insuficiente para Projetos Complexos.
    - Problema: Estimativas não capturam adequadamente esforços de coordenação e dependências.
    - Impacto: Imprecisão em projetos de larga escala (ex: transformações ERP).
    - Evidência: Estudos de caso mostram necessidade de mitigações específicas para projetos complexos [155].
  - Falta de Ferramentas de Operacionalização.
    - Problema: Ausência de ferramentas que incorporem conhecimento de especialistas.
    - Impacto: Dificuldade em aplicar conhecimento acumulado de forma sistemática.
    - Evidência: Necessidade documentada de ferramentas que exponham cost drivers e visualizem impactos [155].
- Lacunas Tecnológicas
  - Precisão Limitada de Modelos Automatizados.
    - Problema: Técnicas de ML/DL não superam significativamente abor-

dagens estatísticas tradicionais.
    * Impacto: Investimento em automação sem retorno proporcional em precisão.
    * Evidência: Replicações mostram que Deep-SE supera *baselines* apenas em poucos casos estatisticamente.significativos [70].
  – Risco de Vazamento de Dados.
    * Problema: Muitas abordagens automatizadas sofrem de data leakage e validação indireta.
    * Impacto: Resultados inflados e pouco confiáveis.
    * Evidência: Revisões sistemáticas destacam riscos de validação inadequada [155].
  – Conjuntos de Dados Desatualizados
    * Problema: Modelos treinam em bases públicas antigas com pouco contexto.
    * Impacto: Redução da transferibilidade e validade externa.
    * Evidência: Falta de metadados e cost drivers em conjunto de dados públicos [107][70].
  – Déficit de Explicabilidade
    * Problema: Métodos ML não justificam estimativas nem facilitam discussão colaborativa.
    * Impacto: Baixa adoção por falta de transparência.
    * Evidência: Necessidade documentada de agentes que combinem explicabilidade com interação humana [155].
- Lacunas Contextuais
  – Informação Limitada em Fases Iniciais
    * Problema: Natureza iterativa e especificação reduzida tornam estimativas precoces complexas.
    * Impacto: Imprecisão sistemática no início dos projetos.
    * Evidência: Gap recorrente na literatura sobre early estimation [107] [70].
  – Variação de Equipe e Competências

    - Problema: Competência e experiência da equipe são drivers críticos não bem modelados.
    - Impacto: Dificuldade em criar modelos universais.
    - Evidência: Diferenças na equipe explicam parte significativa da variação nas estimativas [43]. [155].
  - Limitação por Setor e Escopo
    - Problema: Projetos com requisitos integrados ou domínio específico impõem impactos não capturados.
    - Impacto: Redução da aplicabilidade *Cross-Project* .
    - Evidência: Projetos ERP corporativo mostram dependências não modeladas por técnicas padrão [155].
  - Gestão de Mudanças Frequentes
    - Problema: Mudanças contínuas nos requisitos não são bem incorporadas.
    - Impacto: Estimativas tornam-se rapidamente obsoletas.
    - Evidência: Mudanças de negócio citadas como causa principal de imprecisão [43].

Outras lacunas críticas identificadas:

- Precisão Insuficiente: Métodos atuais (manuais e automatizados) não atingem níveis de precisão adequados para uso confiável.
- Validação Empírica Fraca: Falta de estudos rigorosos de replicação e validação *Cross-Project* .
- Dependência Humana Excessiva: Sobrecarga de métodos subjetivos sem suporte automatizado eficaz.
- Inadequação para Escala: Limitações em projetos complexos e de grande escala.
- Falta de Contextualização: Modelos não capturam adequadamente variações contextuais.
- Ausência de Ferramentas Práticas: Lacuna entre a pesquisa acadêmica e ferramentas utilizáveis na prática.

Como resultado da revisão podemos destacar resumidamente as oportunidades

para pesquisa como segue:

- a) Subexploração de atributos linguísticos explícitos (legibilidade, subjetividade, sentimento) como preditores ou características auxiliares.
- b) Uso incipiente e não sistemático de LLMs, tanto para predição direta de *Story Points* (com protocolos *zero/few-shot/fine-tuning*) quanto para tutoria textual integrável ao processo de escrita e estimativa.
- c) A criação de conjunto de dados contextualizados e atualizados; e a Integração de explicabilidade em modelos automatizados
- c) Limitação semântica em representações clássicas (BoW/TF–IDF), que não capturam o contexto e a pragmática da *User Story*.
- d) O desenvolvimento de métricas de acurácia mais robusta.
- e) Ferramentas que combinem automação com expertise humana.

Das lacunas resumidas acima, aquelas em a), b), c), d) e e) serão tratadas nos próximos capítulos desta tese.

- Investigar atributos linguísticos (legibilidade, subjetividade e sentimento) como características preditivas comparadas a *baselines* consolidadas (por exemplo, TFIDF-SE).
- Investigar o uso de LLMs para predição de *Story Points* com estratégias de *fine-tuning*, em protocolos reproduzíveis e comparáveis.
- Propor um tutor baseado em LLMs para aprimorar a redação de *User Stories*, visando reduzir a ambiguidade e, indiretamente, o erro de estimativa.

3.4 Originalidade

A originalidade da Tese reside na sua abordagem abrangente de comparação de diferentes paradigmas de aplicação de Large Language Models (LLMs) para a estimativa de Story Points.

A seguir, quatro pontos que suportam a originalidade desta pesquisa de doutorado:

- i) Investigação de LLMs como Alternativa: O estudo investiga a eficácia dos LLMs na estimativa de Story Points, oferecendo uma alternativa à maior parte

dos estudos anteriores que focavam principalmente em abordagens de aprendizado de máquina. Pesquisadores têm buscado verificar se os LLMs poderiam aprimorar a predição de *Story Points* a partir da descrição textual das *User Stories*, em comparação com métodos de estimativa tradicionais.

- ii) Comparação Abrangente de Paradigmas de LLMs: Os experimentos ampliam a análise ao comparar diferentes paradigmas de aplicação de LLMs — incluindo fine-tuning, few-shot e zero-shot. Estudos anteriores, como o GPT2SP [46], estavam restritos apenas ao *Fine-Tuning* supervisionado e não exploraram as estratégias zero-shot ou few-shot.
- iii) Contraste com *baselines* Múltiplas: A tese contrasta o desempenho preditivo do modelo LLM ajustado com três baselines:
  - Um modelo preditivo tradicional baseado em vetores TF-IDF acoplados a um classificador de Regressão Linear.
  - Um modelo LLM Zero Shot.
  - Um modelo LLM Few Shot.
- iv) Modelo *Cross-Project*: O modelo ajustado (fine-tuning) foi criado como um modelo único *Cross-Project* (treinado com dados de todos os projetos do conjunto de dados), o que sugere uma maior robustez na generalização entre diferentes domínios, contrastando com os modelos few-shot, zero-shot e TF-IDF que foram treinados individualmente por projeto.
- v) A originalidade do Conjunto de dados produzido (o NeoDataset) reside no fato de ter sido minerado da ferramenta GitLab, enquanto a maioria dos estudos anteriores utilizou dados extraídos do Jira. Além disso, o NeoDataset inclui projetos adicionais que não foram considerados em estudos anteriores, oferecendo uma base de dados mais ampla e diversificada. Diferentemente de outros trabalhos, que compartilham apenas os dados utilizados em seus estudos, o NeoDataset disponibiliza todos os dados coletados, promovendo maior transparência e reprodutibilidade científica.

Essas contribuições, combinadas a um conjunto de dados públicos e a protocolos de avaliação descritos no restante do manuscrito, reforçam o nível de originalidade e a relevância prática da tese.

3.5 Considerações finais

Os trabalhos relacionados demonstram uma evolução substancial no uso de AM/PLN para a estimativa de esforço baseada em *User Stories*. Persistem, entretanto, lacunas de representatividade semântica, exploração de atributos linguísticos explícitos e avaliação sistemática de LLMs. As abordagens sugeridas (discutidas nos próximos capítulos) foram desenhadas para abordar esses pontos, fornecer evidências empíricas comparáveis e apoiar a tomada de decisão em equipes ágeis.

Capítulo 4

Metodologia

Somos autómatos em três quartas partes das nossas ações.

Wilhelm Leibniz

4.1 Introdução

Este capítulo apresenta a metodologia geral adotada na pesquisa, detalhando as decisões, ferramentas, algoritmos e parâmetros que orientaram cada etapa do processo. Justifica-se a escolha das metodologias sob uma perspectiva técnica e científica, seguindo práticas de Opens Science [152] e assim, assegurar a reprodutibilidade e a validade dos resultados. O objetivo é evidenciar como as decisões convergiram para responder às questões de pesquisa e testar as hipóteses.

4.2 Delineamento Geral da Pesquisa

A abordagem metodológica é de natureza quantitativa, explicativa e experimental, possuindo um caráter aplicado. As etapas foram organizadas em um *pipeline* iterativo composto por:

- Revisar a literatura para levantar o estado da arte (Apêndice C);
- Utilizar alguns conjuntos de dados da literatura TAWOSIDatasete[150] e, para diminuir o problema de instabilidade da conclusão, construir um novo conjunto

de dados denominado NeoDataset (Capítulo 5). Criar conjuntos de dados é necessário para diversos fins, e os dados são importantes para o treino de modelos de aprendizagem de máquina e LLM, como, por exemplo, Lira et al. [77].

- Desenvolvimento de três experimentos para validar as hipóteses:
  - **User Story Tutor** — Aplicação de estimativa e recomendação textual com LLMs, utilizando modelo preditivo SVR e TFIDF, que é o *baseline* classico para este tipo de problema [10, 150];
  - **Effort Model** — modelo baseado em Aprendizagem de Máquina com atributos linguísticos usando SVR, para confirmar a hipótese de que os atributos de legibilidade são bons preditores para estimativa de esforço;
  - **Neo LLM Predictor** — modelo de predição direta de *Story Points* por meio de ajuste fino em modelos LLM em comparação com o modelo preditor *baseline* utilizando Regressão Linear no Conjunto de dados TAWOSIDataset;
- Avaliação comparativa com métricas quantitativas (MAE) e instrumentos de avaliação de software (TAM e AttrakDiff).

Cada etapa é descrita a seguir, com suas justificativas e parâmetros específicos.

4.3 Porque utilizar dois conjuntos de dados

Quando utiliza-se mais de um conjunto de dados nos experimentos, evita-se os vieses existentes nos dados e minimizamos o problema da *instabilidade nas conclusões* (mais sobre instabilidade na conclusão na seção 2.5.6). Portanto, a utilização de dois conjuntos de dados na tese teve motivações tanto científicas quanto práticas, visando superar limitações e aumentar a robustez e a validade dos experimentos. Esta tese utilizou dois conjuntos de dados principais:

- NeoDataset: Um novo conjunto de dados construído especificamente para esta pesquisa. Ele foi coletado a partir de projetos reais hospedados no GitLab e contém 40.022 *User Stories* com um total de 163.897 Story Points. Este conjunto de dados serviu de base para os experimentos, incluindo o *Neo*

Legibility Effort Model e o *User Story Tutor*.

- TAWOSIDataset: Este conjunto de dados foi utilizado no experimento de Estimativa com LLM (Capítulo 8). Ele contém 23.313 *User Stories* de 16 projetos diferentes, extraídas do Jira. Ele foi escolhido para este experimento para que o modelo LLM pudesse ser comparado com as *baselines* existentes na literatura e para mitigar a instabilidade dos resultados.

Os principais motivos para utilizar 2 conjuntos de dados foram:

- Minimizar a Instabilidade da Conclusão e Viéses: O uso de múltiplos conjuntos de dados é uma estratégia adotada para minimizar o problema da instabilidade da conclusão (conclusion instability). Esse problema ocorre quando pequenas alterações nos dados de entrada podem levar a conclusões divergentes. Ao realizar replicações experimentais com diferentes bases de dados, a tese buscou evitar vieses existentes nos dados e fortalecer a validade externa dos achados, assegurando uma maior generalização dos resultados.
- Redução de Custo Computacional e Otimização: A escolha do TAWOSIDataset foi motivada por questões de infraestrutura e custo nos experimentos que envolviam modelos de linguagem de larga escala (LLMs). Pois o treinamento de modelos LLM via *Fine-Tuning* demanda um poder computacional considerável. O NeoDataset contem muito ruído (dados de projetos reais em estado bruto no texto) e implica um custo mais elevado para o treinamento. Portanto, para reduzir o custo computacional dos experimentos com LLM, foi utilizado o conjunto de dados TAWOSIDataset.

4.4 Porque usar dois algoritmos nos experimentos

A tese utilizou diferentes algoritmos, como Support Vector Regression (SVR) e Regressão Linear (RL), em suas abordagens por motivos metodológicos e de comparação de desempenho, buscando validar a eficácia de novas abordagens em relação a *baselines* estabelecidos e considerando o custo computacional. Além disso foi utilizado as bibliotecas Python: `scikit-learn`, `NumPy`, `Matplotlib`.

Esses algoritmos testados (SVR e RL) são amplamente empregados em tare-

fas de previsão e estabelecem comparações diretas com os *baselines* da literatura (TFIDF-SE de Porru et al. [111]).

O SVR foi adotado como referência principal, mas também foi utilizada a Regressão Linear para os *baselines* na comparação com o modelo LLM (Capítulo 8).

4.4.1 SVR (Support Vector Regression)

O algoritmo Support Vector Machine (SVM) e sua variante para regressão, SVR (Support Vector Regression), foram utilizados devido ao seu papel como um modelo preditivo consagrado (ou baseline) na literatura de estimativa de esforço a partir de *User Stories*.

- A revisão sistemática da literatura (disponível no Apêndice C) identificou que as técnicas de Term Frequency–Inverse Document Frequency (TF-IDF) combinadas com SVM (ou SVR) são populares para a estimativa de esforço.
- O Effort Model (Capítulo 7) utilizou o SVR como algoritmo principal para prever *Story Points* a partir de atributos linguísticos (legibilidade, sentimento e subjetividade), comparando-o diretamente com o *baseline* TFIDF-SE (que também usa SVM) e a média dos Story Points.
- O módulo Preditor do *User Story* Tutor (UST) (Capítulo 6) também utilizou o SVR (SVM) em combinação com TF-IDF para realizar a estimativa de esforço.

A escolha do SVR deu-se por sua relevância histórica e eficácia comprovada como um dos *baselines* mais fortes para a tarefa de estimativa de Story Points.

4.4.2 Regressão Linear (RL)

A Regressão Linear (RL) foi utilizada principalmente nos experimentos que envolviam Modelos de Linguagem de Grande Escala (LLMs), substituindo o SVR neste contexto por razões de custo computacional e para estabelecer um *baseline* tradicional adicional. No Capítulo 8, a Regressão Linear foi combinada com TF-IDF (Pipeline TF-IDF-RL) para servir como um dos modelos baseline.

- A justificativa para usar a RL, que é menos custosa que o SVR neste contexto, é que se busca reduzir o custo computacional dos experimentos.

- Além disso, a Regressão Linear é um modelo amplamente empregado em tarefas de regressão e estabelece comparativos diretos com *baselines* da literatura.
- O modelo LLM ajustado (fine-tuning) foi comparado com o TF-IDF + RL (além de LLM few-shot e zero-shot).

A Regressão Linear foi introduzida no contexto dos experimentos com LLMs para fornecer uma *baseline* tradicional menos custosa em termos de poder computacional, permitindo comparações eficientes.

4.5 Coleta e Construção do Conjunto de Dados

Para a coleta e construção do conjunto de dados, foi utilizado o Python 3.12, com as bibliotecas `requests`, `CSV` e a API pública do GitLab. Python, por ser uma linguagem bastante utilizada em aprendizado de máquina, conta com uma comunidade atuante e uma extensa biblioteca. Já os dados do GitLab não são utilizados com muita frequência. Optou-se por migrar os dados do GitLab para o formato CSV, pois é um formato bastante conhecido e econômico para a representação de dados.

O GitLab foi escolhido por permitir a extração de *User Stories* e respectivos *Story Points* de forma estruturada, via API oficial, garantindo a reprodutibilidade e a integridade dos dados. O uso de projetos públicos assegura transparência e diversidade nos dados.

Parâmetros adotados

- Seleção de projetos ordenada pelo maior número de estrelas;
- Armazenamento original em formato JSON.
- Conversão para o formato CSV.
- O conjunto de dados final contou com 34 projetos e 40.022 *User Stories*.

Disponibilidade dos artefatos

- O código Python do extrator está disponível em `https://github.com/giseldo/neo-gitlab-extractor`;
- O conjunto de dados produzido (com vários CSVs e JSONs) está disponível em `https://github.com/giseldo/neodataset`;
- O conjunto de dados disponível no HuggingFace pode ser acessado em um único arquivo CSV no endereço: `https://huggingface.co/datasets/giseldo/neodataset`.
- O conjunto de dados também está disponível no Mendeley Data `https://data.mendeley.com/datasets/skk2wn9j86/1`.

4.6 Extração de Atributos e Pré-processamento

Ferramentas: `NLTK`, `TextBlob`, `textstat` e `scikit-learn`.

Justificativa: Essas bibliotecas permitem a extração automatizada dos principais índices de legibilidade (Gunning Fog, Flesch, Dale–Chall, etc.) e medidas de polaridade e subjetividade, amplamente validadas em pesquisas de PLN.

Etapas de pré-processamento

- Normalização textual (remoção de *stopwords*, lematização e *lowercasing*);
- Cálculo dos índices de legibilidade e sentimento;
- Escalonamento dos dados com `StandardScaler`.
- Modelo estimador SVR com TF-IDF.

Parâmetros principais

- Proporção treino/teste: 70/30;
- `random_state` = 42;
- Remoção de *outliers* com base em 1,5×IQR.

4.7 User Story Tutor

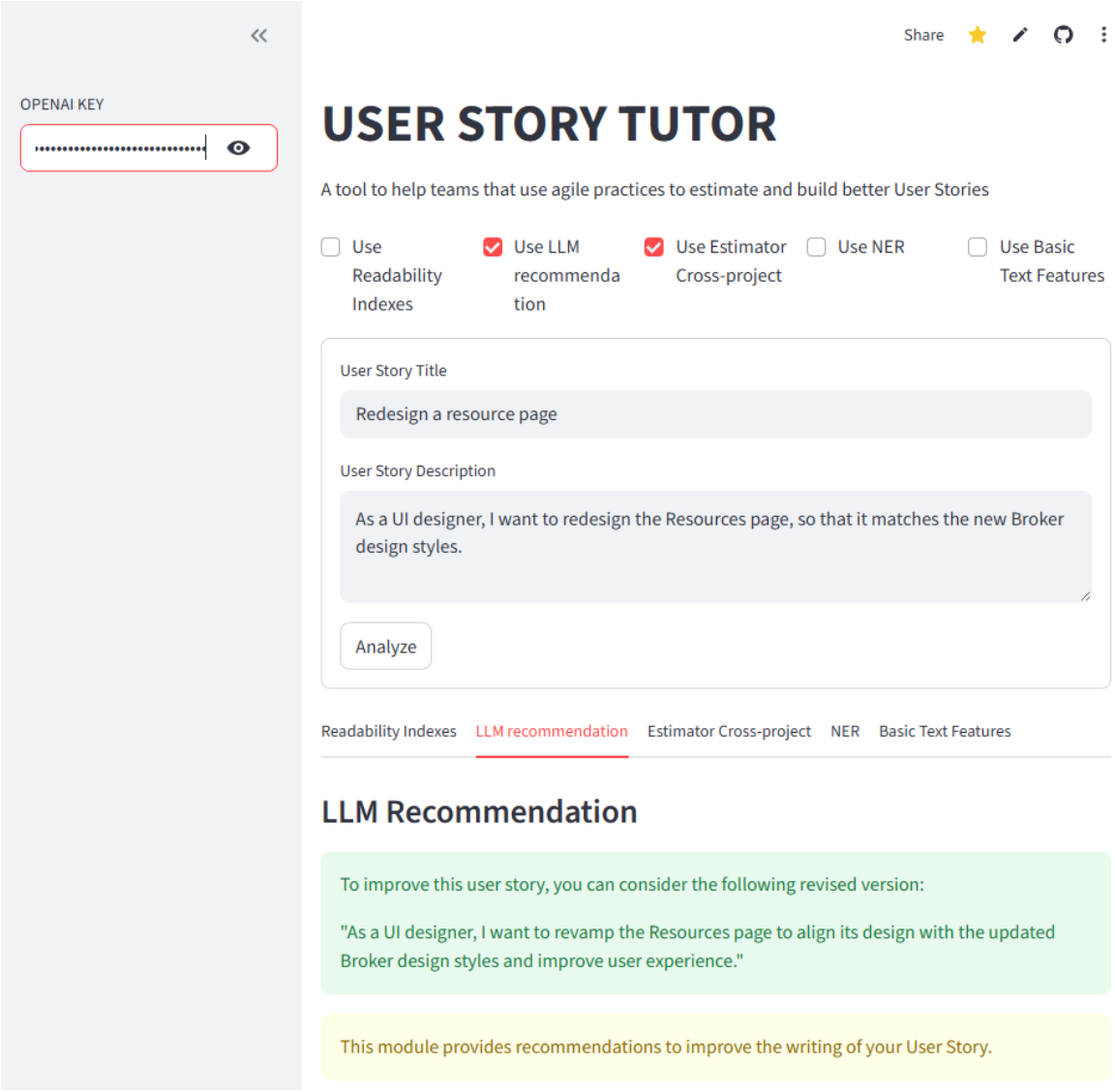
Ferramentas: `Streamlit`.

Aplicação de estimação *Cross-Project* e recomendação textual de user storys com LLMs. Na Figura 4.1 é apresentado a interface do aplicativo, nela o usuário entra com o título e o texto da User Story e o sistema recomenda uma melhoria no texto. Na tela també é possível visualizador a opçao para informar a chave da OpenAI.

Disponibilidade dos artefatos

- Código fonte do projeto `https://github.com/giseldo/userstory`
- Deploy da aplicação on-line `https://userstoryteach.streamlit.app`

Figura 4.1: Tela do *User Story* Tutor hospedada no Stream Lit Cloud



4.8 Modelos com LLMs

Modelos testados: `distilbert-base-uncased`, `Gemma 3:4B`.

O modelo `DistilBERT` foi escolhido por seu equilíbrio entre desempenho e custo computacional, sendo adequado para ajuste fino em GPU de 6 GB. Os modelos `Gemma 3:4B` e foram empregados para inferência *zero-shot* e *few-shot*.

O modelo ajustado foi hospedado no Huggging Face e pode ser utilizado por outros pesquisadores ou empresas interessadas. Veja na Figura 4.2 a tela do modelo ajustado com algumas informações, tais como o tamanho de parâmetros do

modelo 67M de parâmetros, uma versão quantizada menor e o conjunto de dados utilizado. Na Figura 4.3 é apresentado os arquivos que constituem o código fonte do modelo llm ajustado.

Parâmetros de treinamento

- `batch_size` = 8;
- `epochs` = 4;
- `learning_rate` = 2e-5;
- `evaluation_strategy` = epoch;
- `metric` = MSE.

Figura 4.2: Tela do Huggging Face do Modelo LLM ajustado na Tese

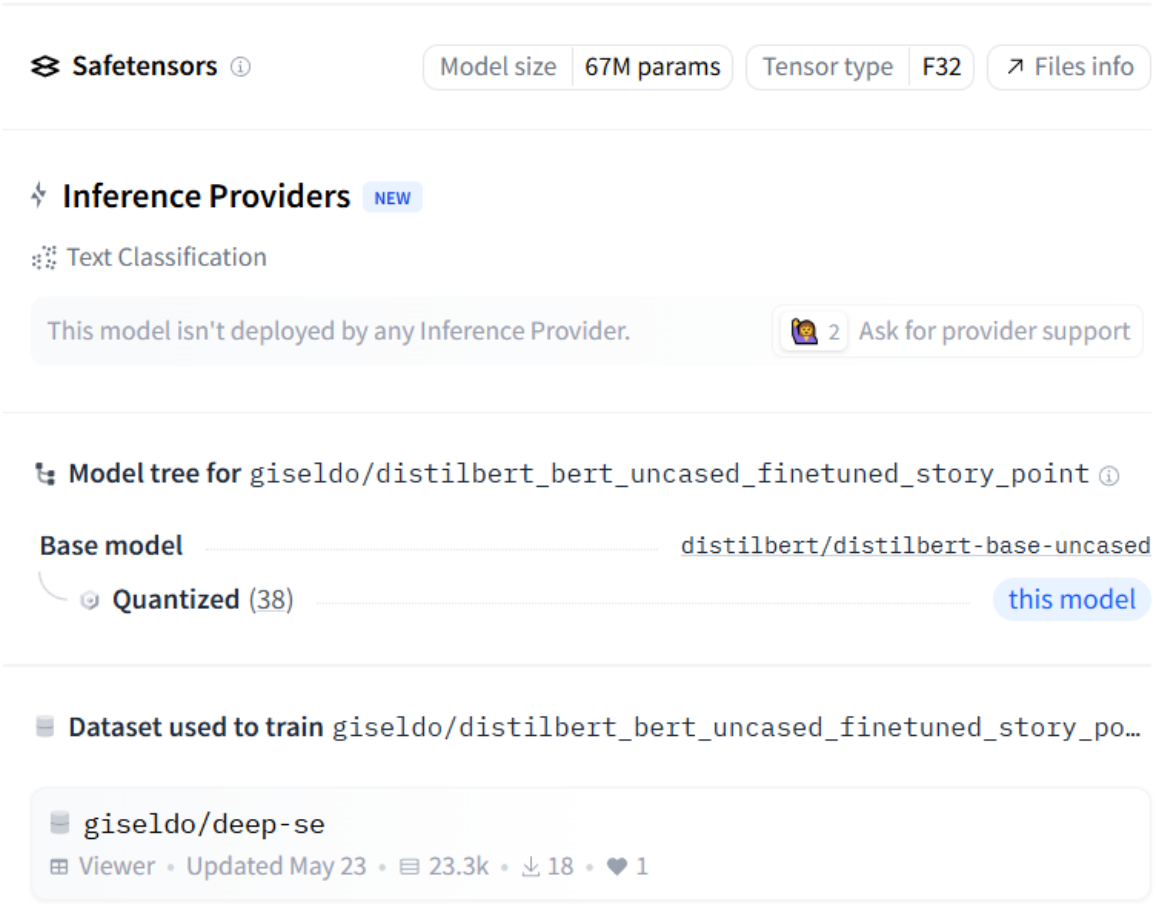
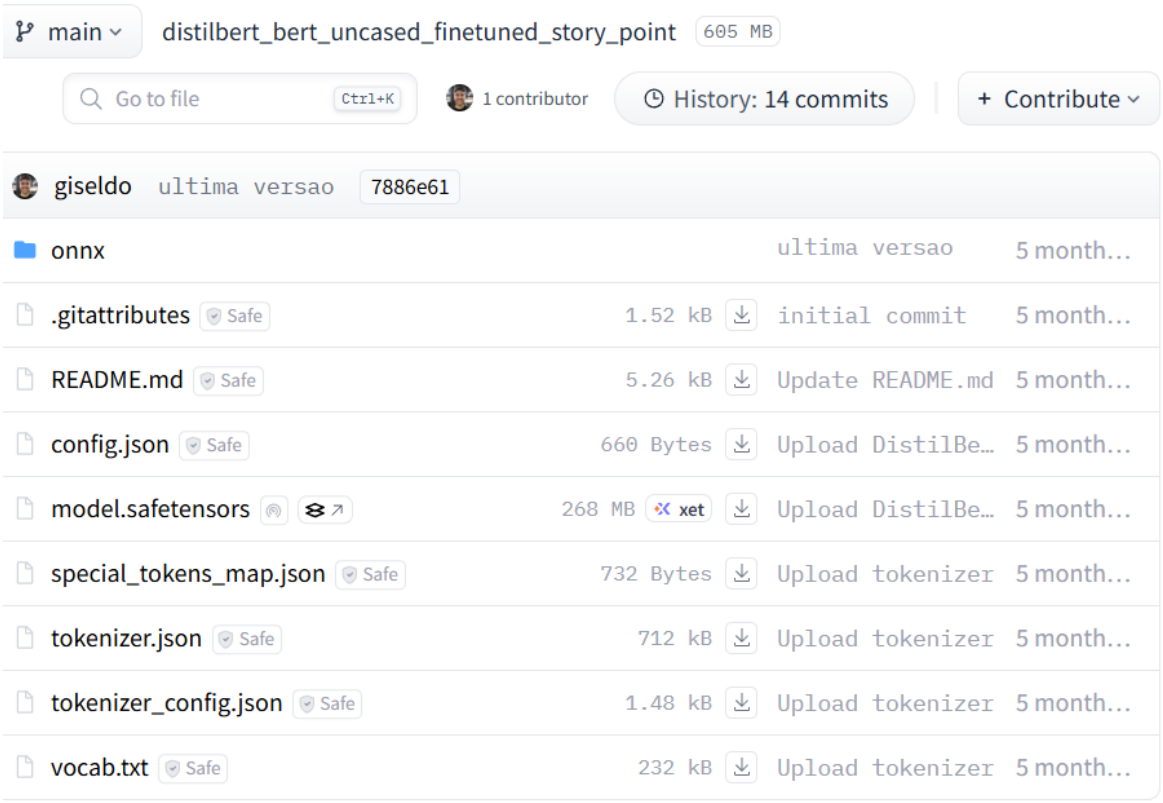


Figura 4.3: Arquivos do LLM Ajustado e quantizado



Em resumo, as ferramentas utilizadas são apresentadas na Tabela 4.1

Tabela 4.1: Ferramentas e justificativas técnicas

Categoria	**Ferramenta**	**Justificativa**
Linguagem	Python 3.12	Ecossistema consolidado para AM e PLN
IDE	VSCode / JupyterLab	Suporte a depuração e execução interativa
Biblioteca ML	`scikit-learn`, `transformers`	Padrão em experimentos reprodutíveis
Visualização	`matplotlib`, `seaborn`	Análise gráfica de desempenho e correlação
Gerência de Dados	`pandas`, `json`	Manipulação e serialização estruturada
Infraestrutura	i3 sétima geração com GPU RX580 4GB	Computador pessoal disponível

Código-fonte que gera o llm ajustado

O código apresentado na Lista I.4 implementa o pipeline de treinamento supervisionado utilizando o modelo DistilBERT para prever *Story Points* a partir de textos de *User Stories*. Inicialmente, são importadas as bibliotecas essenciais - *transformers*, *datasets*, *pandas* e *scikit-learn* - e definidas funções auxiliares (`remove_html_tags` e `remove_urls`) responsáveis por limpar o texto de marcas HTML e URLs. Em seguida, o dataset `deep-se.csv` é carregado a partir do Hugging Face, removendo registros inválidos, como aqueles com `storypoints` iguais a zero ou com valores ausentes. O texto é, então, concatenado em uma nova coluna denominada `context`, que combina o título e a descrição de cada história.

O conjunto de dados é dividido em partes de treino e teste por meio da função `train_test_split`. Ele é convertido para o formato da biblioteca `datasets`, o que permite o uso direto com modelos da Hugging Face. Em seguida, aplica-se o *tokenizer* do modelo `distilbert-base-uncased` a cada exemplo, realizando o truncamento e o preenchimento até 128 tokens. Os rótulos (`labels`) são definidos como valores numéricos contínuos, o que caracteriza o problema como uma tarefa de regressão.

O modelo é instanciado com a classe `AutoModelForSequence..`, configurando `num_labels=1`, o que é apropriado para prever valores contínuos de esforço. As configurações de treinamento (`TrainingArguments`) especificam parâmetros como taxa de aprendizado, número de épocas, tamanho do lote e métricas de avaliação — neste caso, o erro quadrático médio (MSE). Por fim, um objeto `Trainer` é criado para gerenciar o processo de treinamento e avaliação do modelo, que é ajustado por meio do método `train()` e salvo no diretório `"./story_point_predictor"`. A Figura 4.4 apresenta o resultado da execução do Modelo.

Figura 4.4: Resultado da execução do Treino do Ajuste Fine do Modelo Distilbert

Epoch	Training Loss	Validation Loss	Mse
1	140.177900	102.263214	102.263214
2	103.960600	90.528061	90.528053
3	64.470400	88.363815	88.363815
4	57.710200	86.906853	86.906853

Disponibilidade dos artefatos

- O código-fonte para a reprodução dos experimentos está disponível em *https://github.com/giseldo/artigo-storypoint-deep-se-llm*.
- O conjunto de dados utilizado está em *https://huggingface.co/datasets/giseldo/deep-se*.
- O modelo ajustado está disponível no endereço *https://huggingface.co/giseldo/distilbert_bert_uncased_finetuned_story_point*.

4.9 Baselines Utilizados

Os *baselines* constituem um ponto de referência para avaliar o desempenho de novos modelos de estimativa de esforço em projetos ágeis. Nesta tese, os *baselines* foram escolhidos com base na consolidação científica e na ampla adoção na literatura, o que permite comparações alinhadas com práticas comuns e reprodutíveis entre os métodos propostos e os modelos já consagrados.

A estimativa de esforço em *Story Points* a partir do texto da *User Story* é um desafio, e a utilização de *baselines* (modelos de referência) é essencial para validar e comparar o desempenho de novos modelos preditivos.

Os *baselines* utilizados se dividem principalmente em técnicas tradicionais de Aprendizagem de Máquina (AM) baseadas em Processamento de Linguagem Natural (PLN) e abordagens simples (heurísticas ou estatísticas), além de *baselines* específicos para a avaliação de *Large Language Models* (LLMs).

O principal objetivo de se utilizar esses *baselines* é testar se os experimentos realizados conseguem atingir uma precisão comparável ou superior à dos modelos já consolidados.

4.9.1 TFIDF-SE

O modelo **TFIDF-SE**, proposto por Porru et al. [111], é reconhecido como um *baseline* para a tarefa de previsão de *Story Points* a partir do texto das *User Stories*.

Este modelo combina a técnica *Term Frequency–Inverse Document Frequency* (TF–IDF) para representação textual com o algoritmo *Support Vector Machine* (SVM) para predição. A técnica utiliza o TF-IDF para representar o texto da User Story de forma quantitativa. A vetorização transforma o texto em uma matriz numérica, ponderando a importância dos termos no corpus.

Sua escolha como referência deve-se ao fato de ter sido reproduzida e validada em diversos estudos subsequentes, como Choetkiertikul et al. [21] (modelo DEEP-SE) e Tawosi et al. [148], consolidando-a como um padrão comparativo de desempenho.

O TFIDF-SE serviu como referência inicial, superando estimativas simples (como a média ou a mediana). Mesmo após a introdução de modelos de *Deep Learning*, estudos de replicação reafirmaram o TFIDF-SE como um *baseline* sólido.

O *Neo Legibility Effort Model* e o *Neo LLM Predictor* utilizaram o TFIDF-SE (ou variações com TF-IDF) em suas comparações para medir o desempenho preditivo.

4.9.2 Support Vector Regression (SVR)

O Support Vector Regression (SVR) foi adotado como o algoritmo principal nos experimentos que compõem os modelos *Neo Legibility Effort Model* e *User Story Tutor*.

O SVR é a variante de regressão do SVM e figura na literatura como um modelo robusto e amplamente empregado em tarefas de previsão numérica. Sua seleção baseou-se em dois fatores: (i) sua presença recorrente em estudos de estimativa de

esforço (Porru et al. [111]; Moon et al. [88]) e (ii) sua capacidade de generalização mesmo com conjuntos de dados heterogêneos.

4.9.3 Regressão Linear (RL)

A Regressão Linear (RL) foi utilizada como *baseline* adicional nos experimentos com Modelos de Linguagem de Grande Escala (LLMs), integrando o pipeline *TF–IDF + RL*. Sua adoção deve-se à simplicidade de implementação e ao baixo custo computacional, aspectos relevantes nos testes com *fine-tuning* e *few-shot learning*. Esse modelo fornece uma comparação clássica com os métodos baseados em aprendizado profundo, permitindo observar ganhos reais em acurácia e custo.

4.9.4 Média dos Story Points (Mean-Based Regression - MbR)

A média dos *Story Points*, também referida como *Mean-Based Regression* (MbR), é o *baseline* mais simples e fundamental, frequentemente utilizado como referência de comparação.

- **Fundamento:** Este *baseline* prevê o esforço de desenvolvimento para uma *User Story* utilizando a média dos valores de *Story Points* do conjunto de treino.
- **Objetivo de Comparação:** Avalia se o modelo de AM ou LLM supera uma previsão simples. Caso contrário, sua complexidade não se justifica.
- **Uso:** O *Neo Legibility Effort Model* comparou explicitamente seu desempenho com a média dos *Story Points* para validar a relevância dos atributos linguísticos.

4.9.5 LLM versus TF-IDF com Regressão Linear (TF-IDF+RL)

Nos experimentos do *Neo LLM Predictor* o modelo ajustado (*fine-tuning*) foi comparado com *baselines* tradicionais menos onerosos e com abordagens de LLM baseadas em inferência (sem treinamento específico).

Justificativa e Discussão: No pipeline do *Neo LLM Predictor*, uma variação

do *baseline* tradicional foi usada: a combinação de TF-IDF com Regressão Linear (RL).

- **Diferenciação:** Utiliza o mesmo método de vetorização, mas substitui o SVR pela Regressão Linear.
- **Motivação:** A RL reduz o custo computacional e mantém a coerência metodológica com a literatura.

4.9.6 Modelos LLM Few-Shot e Zero-Shot

Os LLMs introduzem novas estratégias de estimação que não dependem de *fine-tuning*, mas sim da capacidade de inferência do modelo pré-treinado (como o Gemma3:4b).

- **LLM Zero-Shot:** O modelo realiza a estimativa baseando-se unicamente em uma instrução textual (*prompt*), sem exemplos prévios.
- **LLM Few-Shot:** O modelo recebe exemplos representativos (cerca de 10% do conjunto de treino) no *prompt*, aprimorando sua inferência.

A comparação com as abordagens *few-shot* e *zero-shot* foi essencial para avaliar a eficácia do modelo LLM ajustado, que obteve um menor MAE em diversos projetos, superando os *baselines* de inferência.

4.9.7 Outros Baselines

Além dos principais *baselines*, a revisão sistemática e os estudos relacionados também citaram outras técnicas simples de comparação:

- Mediana: Utilizada como *baseline* simples, similar à média.
- ZeroR: Algoritmo simples usado por Porru et al. [111] como referência.
- Random Guessing: Estratégia de comparação empregada por Scott e Pfahl.

Veja um resumo dos Baselines na Tabela 4.2.

Tabela 4.2: Resumo dos Baselines de Comparação

Baseline	**Tipo de Modelo**	**Técnica de Representação**	**Justificativa Principal**
TFIDF-SE (SVR)	Tradicional/ML	TF-IDF (vetorização)	Baseline consagrado na literatura, usado para aferir se as novas abordagens (ex: Neo Legibility) conseguem superá-lo.
Média dos SP (MbR)	Heurístico/Estatístico	Média simples	Avalia a efetividade mínima da estimativa; um bom modelo deve sempre superar o erro da média.
TF-IDF-RL	Tradicional/ML	TF-IDF (vetorização)	Baseline tradicional de baixo custo computacional para comparação nos experimentos com LLMs.
LLM Zero-Shot	LLM (Gemma3:4b)	Inferência via prompt	Avalia o desempenho do LLM sem treinamento específico, apenas com instrução.
LLM Few-Shot	LLM (Gemma3:4b)	Inferência via prompt + exemplos	Avalia o desempenho do LLM com o benefício de poucos exemplos contextuais.

4.9.8 Discussão e Justificativa

A escolha dos *baselines* nesta pesquisa considerou três dimensões: (i) relevância científica, baseada em citações e replicações na literatura; (ii) viabilidade computacional, devido à necessidade de comparação com modelos mais complexos, como LLMs; e (iii) consistência experimental, garantindo replicabilidade e controle da instabilidade das conclusões.

O *TFIDF-SE* foi mantido como referência principal por sua estabilidade e histórico de validação. O *SVR* serviu como comparador direto em contextos supervisionados, enquanto a *Regressão Linear* foi empregada para estabelecer um limite inferior de desempenho e um referencial de custo computacional. Essa estratégia permitiu comprovar, de forma metodologicamente sólida, que os modelos propostos (*Neo Legibility Effort Model*, *User Story Tutor* e *Neo LLM Predictor*) superam ou igualam os baselines clássicos em termos de erro médio absoluto e interpretabilidade.

4.10 Métricas de Avaliação

A principal métrica adotada foi o Erro Absoluto Médio (MAE), amplamente utilizado na literatura de estimativa de esforço.

Foram também consideradas:

- RMSE (Root Mean Square Error) — penaliza grandes desvios;
- R^2 (Coeficiente de Determinação) — mede explicação da variância;
- MMRE (Mean Magnitude of Relative Error) — permite comparação direta com *baselines* clássicos.

Nos experimentos com LLMs, também serão realizadas análises qualitativas da coerência das respostas.

4.11 Ameaças à Validade

Para garantir a validade interna, todos os experimentos foram executados com `random_state` fixo e amostras estratificadas. A validade externa foi assegurada pela diversidade de projetos e replicações em múltiplas estratégias (atributos clássicos, ajuste fino e inferência).

4.12 Considerações Finais

A metodologia descrita reflete uma estratégia ~~e triangulação~~ entre abordagens clássicas e modernas de aprendizado de máquina. As decisões sobre coleta, ferramentas e algoritmos foram guiadas por critérios de reprodutibilidade, custo computacional e alinhamento com lacunas na literatura. O detalhamento dos parâmetros e ferramentas aqui apresentados facilita a replicação integral dos experimentos, reforçando a transparência e a robustez científica da Tese.

Para efeito de uma visão geral e rápida da estrutura do restante deste documento, a Tabela 4.3 resume o conteúdo de alguns capítulos 6 a 8 e dos apêndice A a C e as Questões de Pesquisa (QP) respondidas em cada um.

Tabela 4.3: Resumo Capítulo e Apêndice

Capítulo ou Apêndice	**Resumo**	**QP tratada**
6 Estimativa de Story Points com TF-IDF e SVR (User Story Tutor)	O capítulo apresenta o desenvolvimento e avaliação da ferramenta User Story Tutor (UST), projetada para auxiliar equipes ágeis na criação e melhoria de User Stories. A UST utiliza aprendizado de máquina (TF-IDF e SVR) para estimar o esforço em Story Points, além de um modelo de linguagem (LLM) para recomendar melhorias no texto das User Stories e calcular índices de legibilidade. A ferramenta foi avaliada por 40 profissionais de engenharia de software utilizando os frameworks TAM e AttrakDiff, obtendo boa aceitação e confirmando sua eficácia. Conluí-se que a proposta é uma solução viável para criação de User Stories mais claras e eficazes, contribuindo para estimativas mais precisas e melhor planejamento em projetos ágeis.	QP 2: Como melhorar o texto das User Stories para que possam ser usadas em estimativas de Story Points com menor erro? Especificamente: QP 2.1: É possível melhorar a estimativa recomendando mudanças no texto da User Story?
7 Estimativa com Legibilidade e SVR (Neo Legibility Effort Model)	O capítulo 7 apresenta o modelo preditivo Neo Legibility Effort Model, desenvolvido para estimar Story Points a partir do título e descrição de User Stories, utilizando atributos como polaridade do sentimento, subjetividade e legibilidade. A metodologia incluiu a criação de 34 modelos individuais, um para cada projeto do NeoDataset, e a comparação de seus resultados com dois baselines: a média dos Story Points e o modelo TF-IDF com SVR. O modelo proposto superou os baselines, demonstrando menor erro absoluto médio (MAE) e maior eficiência em tempo de execução. Apesar dos resultados promissores, o estudo reconhece limitações, como a representatividade do conjunto de dados e a necessidade de explorar mais informações semânticas das User Stories. Os artefatos do estudo foram disponibilizados no GitHub e RPubs para reprodutibilidade.	QP 3: Qual técnica pode ser utilizada para uma estimativa que supere os baselines identificados (menor erro) para prever Story Points a partir do texto das User Stories?. Especificamente a QP 3.1: Quão bem o modelo com extração de atributos performa em comparação com os baselines?
8 Estimativa com LLM ajustado (Neo LLM Predictor)	O capítulo 8 aborda a eficácia dos Modelos de LLMs na estimativa de Story Points em projetos ágeis, comparando diferentes abordagens: fine-tuning, few-shot, zero-shot e TF-IDF com Regressão Linear. Utilizando o conjunto de dados Deep-SE, o modelo ajustado via fine-tuning (distilbert-base-uncased-story-point) apresentou o menor erro médio absoluto (MAE) na maioria dos projetos, destacando-se pela precisão e capacidade de generalização entre diferentes domínios. Apesar de exigir maior custo computacional inicial, o fine-tuning mostrou-se mais eficiente e econômico em larga escala, sendo uma alternativa promissora para melhorar a estimativa de esforço em User Stories. O estudo reforça o potencial dos LLMs como ferramentas complementares no planejamento ágil.	Este capítulo aborda parte da questão de pesquisa QP 3: Qual técnica pode ser utilizada para uma estimativa que supere os baselines identificados (menor erro) para prever Story Points a partir do texto das User Stories?. QP 3.3: O modelo com LLM tem menor erro em comparação com os baselines?
A Neo SP Estimator	O Neo SP Estimator é uma ferramenta que utiliza cinco métodos complementares para estimar Story Points a partir de User Stories: Regras, Aprendizagem de Máquina, BERT Fine-tuned (Neo LLM Predictor), Groq (Llama 3.1) e Grok (xAI). Ele combina abordagens simbólicas, estatísticas e generativas para oferecer estimativas precisas e contextualizadas, com possibilidade de ajuste manual e aprendizado contínuo. A aplicação permite importar dados históricos via CSV, realizar análises de legibilidade e personalizar palavras-chave para adaptar-se ao domínio do usuário. Todos os dados são armazenados localmente, garantindo privacidade. A ferramenta está disponível online e seu código-fonte pode ser acessado no GitHub.	QP 3.3: O modelo com LLM tem menor erro em comparação com os baselines?
B Melhoria do texto nas User Stories e o impacto nas estimativas Cross-Project	O apêndice B analisa o impacto da melhoria textual de User Stories na estimativa de esforço utilizando o modelo Cross-Project. Foram comparadas versões ruins e aprimoradas de dez User Stories, evidenciando que descrições mais claras e objetivas resultam em estimativas de esforço menores e mais consistentes, enquanto textos confusos geram valores superestimados. Os resultados confirmam que a qualidade textual influencia diretamente o desempenho do modelo, destacando a importância de reduzir ambiguidades e delimitar o escopo funcional. O código fonte do experimento está disponível no GitHub.	QP 2: Como melhorar o texto das User Stories para que possam ser utilizadas em estimativas de Story Points com menor erro?
C Revisão Sistemática	O apêndice apresenta uma revisão sistemática sobre o uso de técnicas de Processamento de Linguagem Natural (PLN) na estimativa de esforço em desenvolvimento de software. Foram analisados 12 estudos primários, identificando que as técnicas TF-IDF com SVM são as mais populares, utilizando principalmente o texto de User Stories como artefato preditor. As métricas mais empregadas foram MMRE, MAE e SA. pesar do crescimento da área, há necessidade de mais estudos empíricos para ampliar as contribuições científicas. A revisão seguiu critérios de inclusão e exclusão, utilizando cinco bases.	QP 1: Qual é o estado da arte para prever Story Points a partir do texto das User Stories?

Capítulo 5

Conjunto de dados (NeoDataset)

Os ignorantes afirmam, os sábios duvidam, os sensatos refletem.

Aristóteles

O conteúdo deste capítulo foi publicado em forma de artigo na conferência “XIII Simpósio Brasileiro de Tecnologia da Informação – SBTI” com o título “NeoDataset - Um conjunto de dados com User Stories e Story Points” [95] e posteriormente selecionado pela e publicado na “RMP (Revista dos Mestrados Profissionais)” (disponível no apêndice D)

Resumo

CONTEXTO: As equipes geralmente utilizam ferramentas de gerenciamento para acompanhar as *User Stories* pendentes, controlar seu código-fonte, registrar suas estimativas de esforço e os responsáveis pela abertura e fechamento dos chamados. Essas ferramentas contêm dados que podem ser utilizados em diversas pesquisas na área de engenharia de software. É desafiador encontrar dados para pesquisas, pois as empresas privadas são relutantes em compartilhar suas informações.

OBJETIVO: O objetivo deste capítulo é apresentar um conjunto de dados contendo dados brutos de 34 Projetos de Software Ágil de código aberto, minerados do GitLab [48], totalizando 163.897 *Story Points* e 40.014 Tarefas.

CONCLUSÃO: Foram disponibilizados esses dados publicamente nos formatos CSV e JSON para facilitar seu uso pela comunidade científica interessada. Acredita-se que esse conjunto de dados pode ser utilizado em várias linhas de pesquisa em engenharia de software, incluindo a estimativa de esforço.

5.1 Introdução

Apesar da necessidade de dados consolidados em pesquisas, as empresas privadas são relutantes em divulgá-los. Além disso, nem sempre eles estão em um formato de fácil acesso para os pesquisadores. O processo de consolidação e preparação dos dados, para transformá-los em informações e *insights*, ainda exige considerável esforço, pois geralmente os dados são armazenados em bancos de dados relacionais.

Para contribuir com pesquisas de engenharia de software, e inspirados pela contribuição de outros conjuntos de dados [67, 146], foi disponibilizado um novo conjunto de dados (chamado NeoDataset). Esse conjunto de dados abrange dados de 34 projetos, com 163.897 tarefas (ou *issues*) retirados de repositórios do GitLab [48], totalizando 40.014 Story Points. Ele é disponibilizado no GitHub para que toda a comunidade interessada possa contribuir, semelhante ao que acontece com outros conjuntos de dados [156].

As seções seguintes apresentam uma descrição do conjunto de dados, como ele foi extraído, suas características e estrutura, sua originalidade e relevância, e as considerações finais.

5.2 Justificativa

A criação do NeoDataset foi motivada por um conjunto de lacunas técnicas e científicas identificadas durante a execução da Tese. A principal delas refere-se à escassez de conjuntos de dados públicos e representativos que permitem a reprodução e a comparação de experimentos no domínio da estimativa de esforço em projetos ágeis de software. Conforme apontado por diversos estudos na lite-

ratura, a ausência de dados abertos tem dificultado a consolidação de resultados e contribuído para o problema conhecido como *instabilidade das conclusões*, em que diferentes autores, utilizando metodologias semelhantes, chegam a resultados divergentes em virtude das diferenças nos dados empregados [69, 15].

Além disso, os conjuntos de dados existentes, como os propostos por Porru et al. [111], Choetkiertikul et al. [23], Tawosi et al. [145], apesar de relevantes, são limitados em tamanho, contexto e representatividade. A maioria deles contém dados derivados de experimentos controlados ou dados provenientes de domínios específicos, o que reduz a capacidade de generalização dos modelos de AM e de PLN aplicados à estimativa de esforço. Outro obstáculo recorrente é a dificuldade de acesso a dados reais de empresas.

Diante desse cenário, o NeoDataset foi concebido com o propósito de preencher essa lacuna e oferecer uma base pública, ampla e realista para pesquisas em engenharia de software. Ele foi construído a partir de projetos reais hospedados no GitLab, extraídos via API pública, garantindo integridade, rastreabilidade e diversidade de contextos. Essa estrutura permite a exploração de diferentes tipos de tarefas de AM e PLN, como classificação de texto, predição de valores contínuos e análise de legibilidade.

A disponibilização do NeoDataset também visa atender a princípios de reprodutibilidade e transparência científica. Todo o código utilizado para a coleta e o pré-processamento dos dados foi publicado no repositório GitHub, permitindo que outros pesquisadores possam replicar os experimentos ou estender o conjunto de dados com novos projetos. Essa abertura segue a filosofia de iniciativas anteriores, como o TAWOSIDATASET, promovendo uma cultura de ciência aberta e colaborativa.

Do ponto de vista metodológico, o NeoDataset foi essencial para os experimentos desta tese, especialmente na validação dos modelos *Effort Model* e *User Story Tutor*. A utilização de dados reais e diversificados possibilitou avaliar o desempenho dos modelos em cenários mais próximos da prática industrial, aumentando a validade externa dos resultados obtidos.

Em síntese, a criação do NeoDataset justifica-se por três dimensões comple-

mentares:

- **Dimensão científica:** promover a reprodutibilidade experimental e superar a carência de bases públicas e abertas para estudos de estimativa de esforço;
- **Dimensão técnica:** oferecer um conjunto estruturado e diversificado extraído automaticamente de projetos reais do GitLab, com métricas consistentes e documentação completa;
- **Dimensão prática:** apoiar a comunidade acadêmica e industrial na construção de modelos de estimativa de esforço mais precisos, interpretáveis e aplicáveis à realidade de equipes ágeis.

5.3 Metodologia

O procedimento metodológico é composto por quatro etapas principais. Inicialmente, foram definidos os critérios de seleção dos projetos; estes foram filtrados pelo número de estrelas do repositório. Na segunda etapa, realizou-se a autenticação, e os dados foram extraídos no formato original (JSON). A terceira etapa envolveu o processamento desses dados. Finalmente, na quarta etapa, os dados foram agrupados, resultando na obtenção do arquivo final no formato CSV, pronto para análise ou integração em outras plataformas.

5.3.1 Extração

Este conjunto de dados foi extraído durante os meses de janeiro de 2023 e abril de 2023. O processo de mineração teve como alvo os principais projetos de código aberto do GitLab [48]. Os projetos selecionados empregam metodologias ágeis de desenvolvimento de software e registraram o tamanho em *Story Points* de suas tarefas.

Para minerar informações do GitLab [48] foi criada uma ferramenta de extração implementada em Python que se conecta ao GitLab via *API*, o código dessa aplicação está disponível no GitHub.

Foram coletadas somente *User Stories* com o atributo *State* (situação): *Closed*

e que tenham o atributo *weight* (esse campo, no GitLab [48], é utilizado para registrar o esforço em Story Points) preenchido. Mais informações sobre os projetos incluídos no conjunto de dados estão disponíveis diretamente no *GitLab*.

5.3.2 Armazenamento

O conjunto de dados foi armazenado nos formatos JSON e CSV, dada a simplicidade de lidar com ambos, os quais estão disponíveis no GitHub.

5.3.3 Característica

O conjunto de dados NeoDataset contém um total de 163.897 tarefas oriundas de 34 projetos, totalizando 40.014 *Story Points*. Os projetos possuem características distintas, abrangendo diversas linguagens de programação, diferentes domínios de negócio e localizações geográficas variadas da equipe. A Tabela 5.1 apresenta informações resumidas do conjunto de dados, incluindo o nome do Projeto, o Total de Tarefas e o total de Story Points. A coluna ID da Tabela apresenta o ID do projeto no site do Gitlab [48]. O Total de Tarefas refere-se ao total de *User Stories* de um projeto que se encontra na situação *Closed*. O Total de *Story Points* de um Projeto é o somatório dos *Story Points* de todas as Tarefas informadas pelos usuários desse projeto.

5.3.4 Estrutura

A Tabela 5.2 apresenta uma descrição dos principais atributos da entidade Tarefas; outros campos foram omitidos.

Um exemplo dos dados do conjunto de dados está na Tabela 5.3. Nesta tabela, são apresentados somente os campos título, descrição e Story Point; os demais campos foram omitidos.

A Figura 5.1 mostra um Diagrama de Classes que representa visualmente a estrutura JSON do conjunto de dados. A entidade principal é Tarefa (no diagrama,

Tabela 5.1: Estatísticas descritivas dos dados do NeoDataset.

ID	Projeto	Total Tarefas	Story Points							
			Total SP	Média	STD	Mín	25%	50%	75%	Max
10152778	Minds	521	1594	3,1	4,2	0	1	2	3	80
10171280	Minds Mobile	2796	7213	2,6	2,7	0	1	2	3	50
12894267	MLReef	285	1419	5,0	5,0	0	2	4	6	40
3828396	GitLab Chart	15	31	2,1	1,2	1	1	2	2	5
278964	GitLab	19548	41270	2,1	1,8	0	1	2	3	160
6206924	Tildes	42	91	2,2	1,0	1	1,25	2	3	4
12584701	StackGres	171	1106	6,5	7,4	1	2	4	8	48
7764	Gitlab.com	355	970	2,7	7,2	0	1	2	3	128
12450835	Duplicity	424	5798	13,7	22,1	4	6	6	12	260
28847821	Buyer Experience	1004	2665	2,7	2,0	0	1	2	4	14
10174980	Veloren	178	502	2,8	2,8	1	1	2	3	15
250833	Gitlab Runner	13	35	2,7	2,2	1	1	2	3	9
23285197	Subway	284	507	1,8	1,5	0	1	1	2	13
1304532	Gitlab GL Infra Reliability	1724	4520	2,6	2,5	0	1	2	3	21
14052249	Mythictable	167	509	3,0	2,9	1	2	2	3	20
28419588	Lazarus	144	17800	123,6	48,7	100	100	100	100	300
4456656	Pajamas Design System	344	684	2,0	1,6	0	1	1	3	13
3836952	Tezos	103	2771	26,9	38,6	0	2,5	5	32	100
7603319	Meltano	237	1107	4,7	5,3	0	1	4	8	40
7776928	Triage ops	216	442	2,0	1,2	1	1	2	2	10
2670515	Customers gitlab com	1574	3137	2,0	1,2	0	1	2	2	15
21149814	Opengeoweb	1942	5668	2,9	1,5	1	2	3	3	20
15502567	Kicad	3284	34692	10,6	14,2	4	6	6	10	268
1714548	Petals Vockpit	154	400	2,6	1,5	1	1	2	3	8
7128869	Nlx	327	1355	4,1	3,6	0	2	3	5	21
10171263	Minds Backend Engine	982	3742	3,8	3,7	0	2	3	5	32
14976868	Database Lab Engine	113	721	6,4	7,6	1	2	4	8	42
2009901	Gitaly	171	401	2,3	1,5	1	1	2	3	13
28644964	FPC Source	102	14200	139,2	63,2	100	100	100	200	300
7071551	Gitlab UI	310	600	1,9	2,0	0	1	2	2	32
734943	Gitlab Pages	121	265	2,2	2,6	1	1	1	3	20
10171270	Minds Frontend	1845	5695	3,1	3,2	0	1	2	4	40
5261717	Gitlab vscode extension	106	185	1,7	0,7	1	1	2	2	4
19921167	Glaxnimate	420	1802	4,3	4,0	1	2	3	5	40
	Total	40.022	163.897							

Tabela 5.2: Descrição dos principais atributos.

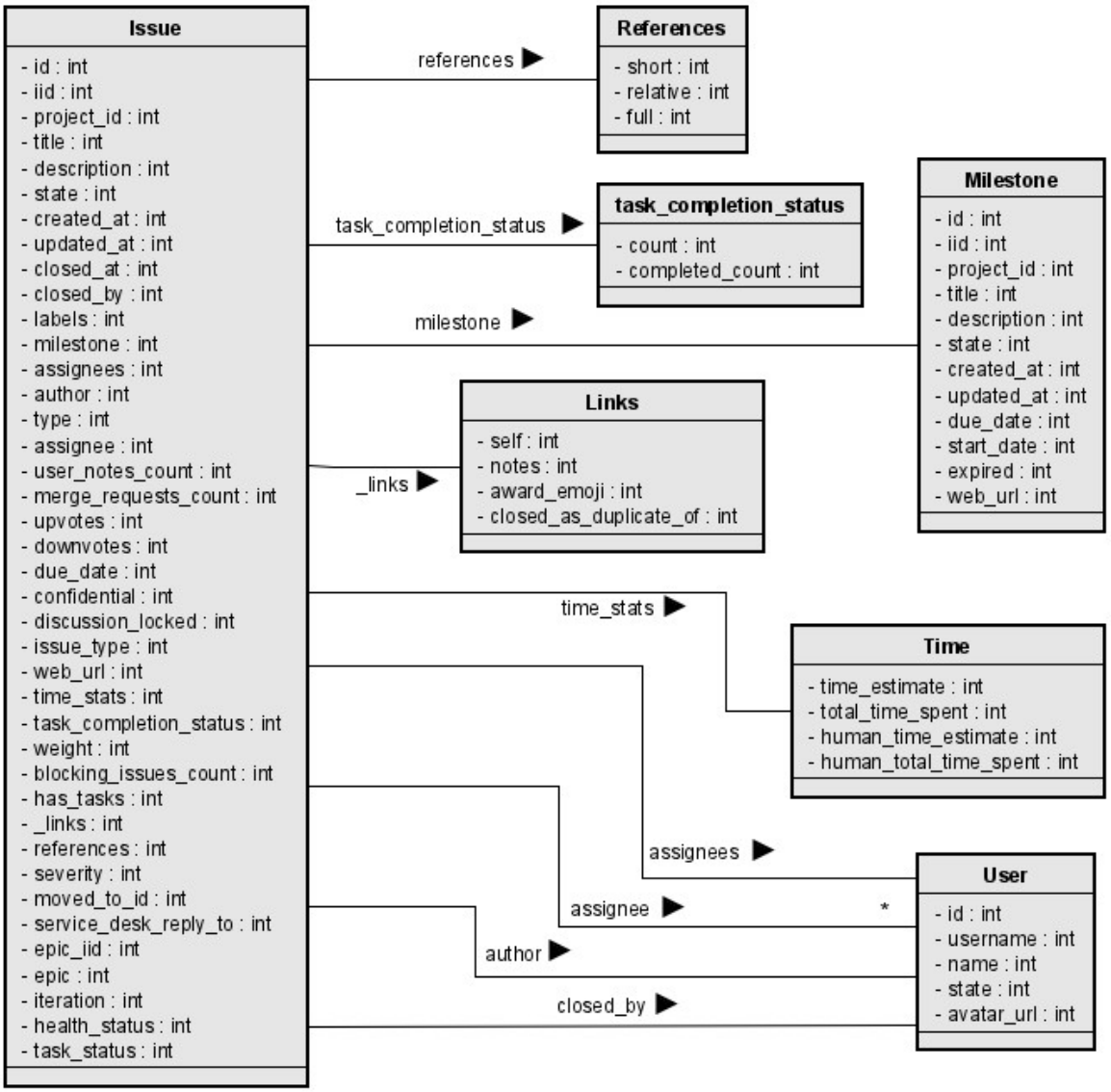
Nome	**Descrição**
Título	Título da *User Story*
Descrição	Descrição da *User Story*
SP (ou Weight)	*Story Points* estimados

Tabela 5.3: Exemplo dos dados.

Título	**Descrição**	**SP**
Posts repeat in Groups feed	We have a few reports of posts repeating in groups. [This Gitlab issue](url...) makes reference to the issue in a test Group. Additionally, a user alerted me that [this Group](url...) exhibits the issue, though notably the posts repeat only after scrolling VERY deep into the timeline (~100+ posts). The pinned comment re-appears in the feed, after which the feed appears to loop. Note timestamps on the posts in this sequence: ![image](url...) ![image](url...) ![image](url...) ### Replication steps Effect can be seen here: (url...) To replicate a base-case: 1. Make a new group 2. Make a post titled 1, and a post titled 2 3. Refresh	5
Expose bot accuracy scores on front end for admins	In order to (a) help with making admin decisions, and (b) QA check the scores so that admins can see what Tasman is spitting out and provide feedback on the effectiveness of the current scoring, we'd like to make Tasman-derived trust scores visible on the front end of Minds website for users with Admin access. (url...) >Designs are in progress - Given Tasman trust scores for users are available, - and UserA is logged into Minds, - and UserA's account has Admin access, - when any user avatar + name component is displayed, - then a number is displayed alongside the avatar + name that exposes the Tasman trust score for that user. Proposed approach for displaying the correct colour based on a 0-100 value ->(url...) ### Notes * Uniquely an admin feature for the first iteration * In the future this will be visible to all users, including an explainer (via a modal) about the score, why scores are valuable and how to improve scores.	3
Refactor experiments to use events instead of contexts	## The Problem Currently all users are bucketed into experiments on app initialisation and not when they actually see the experiment. This was done so that pageviews could register the experiments as contexts when someone lands on the site. Issues arise when experiment reports rely on only including users who have seen the experiment to be included. For example, an experiment that forwards users to discovery instead of the newsfeed after registration should not just only include users that have just signed up, it should also only apply to users that do not have any referral logic (ie. take newly loggedin/registered users back to previous page). ## Proposed change - [x] Create a new iglu schema 'growthbook_event' which includes the experiment_id that the user is in - [x] Remove backend registration and client on init logic - [x] Record a 'growthbook_event' via snowplow when an experiment is run - [x] To avoid unnecessary events, cache a reference to the events so that they only get sent every 24 hours. - [x] Now that we support psuedo_id, attach both the anonymous_id and user_id to growthbook events	8

Issue), que contém as principais informações. O conjunto de dados tem mais de 70 atributos.

Figura 5.1: Diagrama de classes do arquivo JSON representando sua estrutura.



5.4 Originalidade e relevância

O conjunto de dados aqui apresentado foi minerado da ferramenta de gerenciamento *GitLab* [48] e inclui projetos adicionais que não foram utilizados por nenhum desses estudos anteriores. Já existem estudos anteriores que extraíram

dados da ferramenta de gerenciamento Jira [9] para construir modelos preditivos [151, 23, 111, 130], mas projetos extraídos do *GitLab* [48] são mais raros.

Assim como fez [146], compartilho todos os dados coletados. Pois, o mais comum é compartilhar apenas os dados do conjunto de dados considerados no próprio estudo, como fez, por exemplo: [23], e não todos os dados coletados.

A contribuição esperada é que este conjunto de dados possa auxiliar pesquisas na área de ASD. Embora o conjunto de dados tenha sido projetado inicialmente para a pesquisa de estimativa de *Story Points* em ASD, ele também inclui informações relevantes para outras pesquisas de engenharia de software. Além de fornecer uma possibilidade para reproduzir achados de outros estudos.

5.5 Disponibilidade dos artefatos

- O código Python do extrator está disponível em `https://github.com/giseldo/neo-gitlab-extractor`;
- O conjunto de dados produzido (com vários CSVs e JSONs) está disponível em `https://github.com/giseldo/neodataset`;
- O conjunto de dados disponível no HuggingFace pode ser acessado em um único arquivo CSV no endereço: `https://huggingface.co/datasets/giseldo/neodataset` e no Mendeley Data `https://data.mendeley.com/datasets/skk2wn9j86/1`.

5.6 Considerações finais

Nesse capítulo apresentou-se o NeoDataset, um conjunto de dados com projetos minerados do *GitLab* [48], para ser utilizado em pesquisas que exploram vários problemas do desenvolvimento de software ágil. Como tal, ele será utilizado nos próximos capítulos.

Capítulo 6

Estimativa de Story Points com TF-IDF e SVR (User Story Tutor)

Como cientista, eu me sinto apaixonado pela verdade. Eu amo a verdade, ela é tão empolgante.

Richard Dawkins

Este capítulo foi publicado na forma de artigo “CSEDU 2024 - International Conference on Computer Supported Education” com o título “User Story Tutor (UST) to Support Agile Software Developers” [94] (disponível no apêndice E). Além disso, recebeu o prêmio de melhor artigo de estudante no mesmo evento. Uma versão expandida do artigo foi publicada na revista *Springer Nature Computer Science* [29].

Este capítulo responde à questão de pesquisa: “QP 2: Como melhorar o texto das *User Stories* para que possam ser usadas em estimativas de *Story Points* com menor erro? Especificamente: QP 2.1: É possível melhorar a estimativa recomendando mudanças no texto da User Story? A motivação foi melhorar a estimativa em Story Points, recomendando uma melhoria no texto da *User Story* com um LLM.

Resumo

CONTEXTO: As *User Stories* registram o que deve ser construído em projetos que utilizam práticas ágeis. As *User Stories* servem tanto para estimar o esforço,

geralmente medido em Story Points, quanto para planejar o que deve ser feito em uma Sprint. Esta tese tenciona utilizar o texto das *User Stories* para estimar o esforço. É, então, necessário "processar" o texto para utilizar suas características como variáveis de entrada e gerar uma estimativa de esforço (variável dependente na saída). Parece razoável supor que, quanto melhor elaborado for tal texto, melhor será a estimativa. Portanto, é essencial treinar engenheiros de software sobre como criar *User Stories* simples, de fácil leitura e abrangentes.

OBJETIVO: O objetivo é fornecer uma estimativa com aprendizado de máquina e exibir indicadores de legibilidade, além de recomendar melhorias no texto da User Story.

MÉTODO: Foi projetada uma aplicação web chamada *User Story* Tutor (UST) que verifica a legibilidade da descrição de uma determinada *User Story* e, se necessário, recomenda práticas apropriadas para melhoria com LLM. O UST também estima o esforço da *User Story* em *Story Points* usando técnicas de Aprendizagem de Máquina (TF-IDF com SVR). Como tal, o UST pode apoiar a educação contínua de equipes de desenvolvimento ágil ao escrever e revisar *User Stories*.

AVALIAÇÃO: A usabilidade do UST foi avaliada por 40 profissionais com o Technology Acceptance Model (TAM) e o AttrakDiff.

RESULTADOS: A distribuição das respostas da avaliação do TAM foi a desejada em todas as variáveis consideradas. A análise da avaliação com o AttrakDiff também confirmou os resultados do TAM.

CONCLUSÃO: Aparentemente, o UST pode ser usado com boa confiabilidade.

6.1 Introdução

Escrever uma boa *User Story* pode ser difícil. A *User Story* pode ser muito superficial e não apresentar detalhes adequados para a compreensão do resultado esperado, ou, ao contrário, pode ser muito abrangente. Por exemplo, uma parte interessada pode confundir o nível de detalhe de uma *User Story* e escrever o escopo de um módulo ou sistema inteiro, o que não é apropriado. Além disso, a qualidade das *User Stories* pode ter um grande impacto nas estimativas. A escrita de boas

User Stories, que podem ser usadas para estimar esforço, aparece como um dos 5 problemas ágeis mais importantes informados por 119 desenvolvedores [7].

Aprimorar a criação da *User Story* é crucial para um melhor planejamento e, consequentemente, para o sucesso de projetos. Portanto, o objetivo deste capítulo é auxiliar o processo de criação de *User Story*, recomendando melhorias, utilizando processamento de linguagem natural. Uma contribuição esperada é ajudar equipes de desenvolvimento que utilizam práticas ágeis a construir melhores *User Stories*.

A ferramenta UST recebe como entrada o título e a descrição da *User Story* em inglês e apresenta recomendações personalizadas para melhorá-la, com o apoio de um modelo de linguagem LLM. LLMs são modelos treinados em grandes quantidades de dados. A ferramenta também apresenta uma previsão em Story Points, gerada por um algoritmo de aprendizado de máquina treinado com dados de outros projetos. O usuário também visualiza os índices de legibilidade da *User Story*, que podem ser usados para representar um indicador de clareza do texto.

Foi utilizada a metodologia Design Science Research com 3 fases: identificação do problema, desenho da solução e avaliação para construir a ferramenta [169]. Para avaliação, foi realizada uma pesquisa com apoio do framework Technology Acceptance Model (TAM) [34] e do framework de avaliação AttrakDiff [56]. Quarenta profissionais que não participaram do desenvolvimento da UST avaliaram a solução e responderam à pesquisa. Os resultados da avaliação do TAM e do AttrakDiff indicam que o UST atende aos objetivos estabelecidos, com boa aceitação por parte dos participantes.

O restante deste capítulo está organizado da seguinte forma. A seção 6.2 discute a metodologia e os artefatos metodológicos adotados, trazendo mais detalhes técnicos do UST. A seção 6.2 aborda os resultados da avaliação do UST. A seção 6.4 é dedicada a trabalhos relacionados. A seção 6.6 explora ameaças à validade da investigação e seus resultados. Por fim, a seção 6.8 apresenta as considerações finais.

6.2 Metodologia

Esta seção apresenta uma visão geral das tecnologias usadas para construir o UST, sua arquitetura de alto nível, o conjunto de dados utilizado e sua interface de aplicação, além da metodologia.

A metodologia de pesquisa utilizada foi o Design Science Research [169]. Foi projetado um aplicativo da web chamado *User Story* Tutor (UST) que usa processamento de linguagem natural, índices de legibilidade e previsão de aprendizado de máquina como prova de conceito para melhorar a escrita de *User Stories*. Foi utilizada uma pesquisa, apoiada por um questionário no Google Forms, para avaliar a UST. O desenvolvimento foi realizado nas seguintes etapas:

- Revisão da literatura;
- Design de um modelo preditivo que prevê o número de *Story Points* da *User Story* usando aprendizado de máquina;
- Definição dos índices básicos de legibilidade do texto que podem ser extraídos de *User Stories*;
- Design de um módulo de recomendação por meio da consulta ao ChatGPT;
- Implementação de todos os três módulos como uma aplicação web;
- Avaliação interna utilizando diversas *User Stories* de projetos reais;
- Avaliação com participantes em uma pesquisa;
- Análise qualitativa e quantitativa dos resultados da pesquisa.

6.3 Explicação do Modelo de AM

O objetivo do modelo de AM utilizado é prever os valores de *Story Points* (tarefa de regressão) a partir do texto combinado dos campos *title* e *description*. O Modelo utilizou dados *Cross-Project* com SVR e TF-IDF.

O Pipeline é: Pré-processamento (ou limpeza) $\rightarrow$ TF-IDF $\rightarrow$ regressão com SVR. Além disso, O SVR é capaz de modelar relações não lineares (via kernel RBF) sem a necessidade de muitas variáveis explicativas.

Pré-processamento

- Utiliza o *spaCy* (`en_core_web_sm`) para lematização e remoção de *stopwords*.
- Remove URLs e espaços em branco utilizando expressões regulares.
- Mantém apenas tokens alfabéticos que não sejam *stopwords*, aplica lematização e converte para letras minúsculas.

Extração de Características

Os textos pré-processados são transformados em uma matriz Tfidf por meio do TfidfVectorizer, que representa o texto como um modelo bag-of-words ponderado pela frequência e importância dos termos.

Modelo de AM

O modelo utilizado é o **SVR** (*Support Vector Regression*) implementado em `sklearn.svm.SVR`, que utiliza por padrão o *kernel* RBF.

O treinamento é realizado com:

$$X = \text{TF-IDF(atributos)} \quad \text{e} \quad y = \text{storypoints.}$$

Tratamento de Dados

- O conjunto de dados é o NeoDataset lido a partir de um arquivo com (`hf://datasets/...`) hospedado no Hugging Face.
- Linhas contendo valores `NaN` são removidas.
- Outliers nos *story points* são eliminados, mantendo-se apenas os valores entre os percentis de 5% e 95%.

Persistência

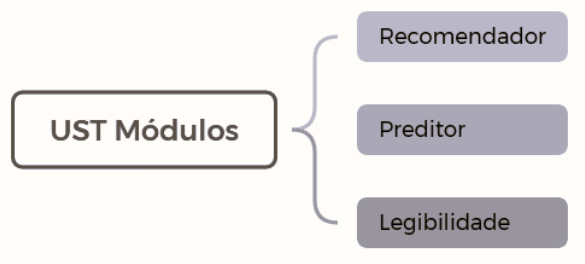
Tanto o vetorizar (`vec`) quanto o modelo treinado (`reg`) são salvos utilizando `joblib` nos arquivos:

- `models/cross_project.vec`
- `models/cross_project.model`

6.3.1 Visão geral

A arquitetura do UST foi dividida em 3 módulos: Recomendador, Estimador e Legibilidade (Figura 6.1), e para cada módulo foram selecionados e executados determinados procedimentos. Esses procedimentos gerais selecionados serão detalhados nas subseções seguintes.

Figura 6.1: Módulos do UST



Recomendador

O módulo Recomendador é responsável por recomendar melhorias nas *User Stories*, retornando texto em linguagem natural. O modelo LLM utilizado (disponibilizado pela OpenAI) já é treinado com uma grande quantidade de texto, extraído de toda a internet [157]. Porém, a customização do prompt foi necessária para melhor personalizar a devolução.

Para customizar a resposta do modelo OpenAI LLM é enviado um prompt para configurar o retorno de acordo com a necessidade do sistema de recomendação. Seguindo as diretrizes da OpenAI para a construção de prompts eficazes, buscou-se uma agilidade clara e precisa para não gerar dúvidas na hora de retornar a recomendação. A probabilidade de uma boa recomendação de retorno depende da

objetividade do prompt oculto enviado junto com a recomendação. Com a intenção de limitar a tarefa, é enviado um prompt para limitar o conjunto de possibilidades de retorno, já que tarefas complexas geralmente têm uma taxa de erro maior do que solicitações de tarefas mais simples. Além disso, várias interações são necessárias para criar o prompt utilizado em busca de melhorias e seguindo as boas práticas recomendadas pela própria OpenAI.

Preditor

O módulo Estimador utiliza um algoritmo de aprendizagem supervisionada, selecionado de acordo com sua capacidade de previsão para prever os *Story Points* a partir do texto da *User Story* informado. Para a elaboração do módulo preditor seguem-se as técnicas: coleta de dados; exploração de dados; preparação de dados; criação, treinamento e validação; ajuste de hiperparâmetros e implementação do modelo.

O preditor *User Story* Estimator foi treinado com outras *User Stories* de outros projetos ágeis. Este conjunto de dados foi coletado de projetos reais de código aberto extraídos de um repositório de código aberto GitLab [48]. A métrica utilizada para seleção do melhor algoritmo e ajuste dos hiperparâmetros foi o Erro Médio Absoluto com validação cruzada. Ao final, o modelo preditor foi treinado com todos os dados e disponibilizado para a aplicação.

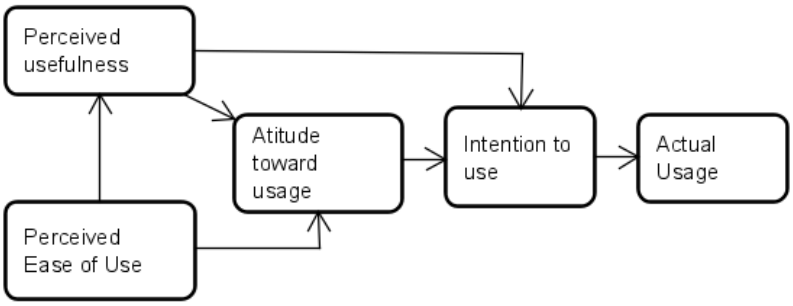
Legibilidade

Por fim, o módulo relacionado à Legibilidade apresenta 4 índices de legibilidade de texto: Gunning Fog, Índice de legibilidade automatizado, Índice Coleman-Liau e Flesch Reading Ease. Mas é importante destacar que, nesta abordagem, para facilitar a interpretação do índice de legibilidade de forma geral, também foi criada uma variável chamada *Resultado Final*, sendo a média aritmética dos 4 índices selecionados.

6.3.2 Avaliação

Para a avaliação do UST, foi realizada uma pesquisa baseada na estrutura do TAM [34] e na estrutura de avaliação AttrakDiff [56]. TAM é muito utilizado em sistemas de informação para verificar como os usuários aceitam e usam a tecnologia – veja a Figura 6.2. Para o teste estatístico TAM, foi utilizado o alfa de Cronbach. O teste AttrakDiff [56] apresenta fatores de qualidade (hedônicos e pragmáticos) que podem ajudar a avaliar melhor a abordagem, complementando o framework TAM. A pesquisa também coletou percepções e sugestões dos participantes em relação à UST.

Figura 6.2: Arquitetura do TAM.



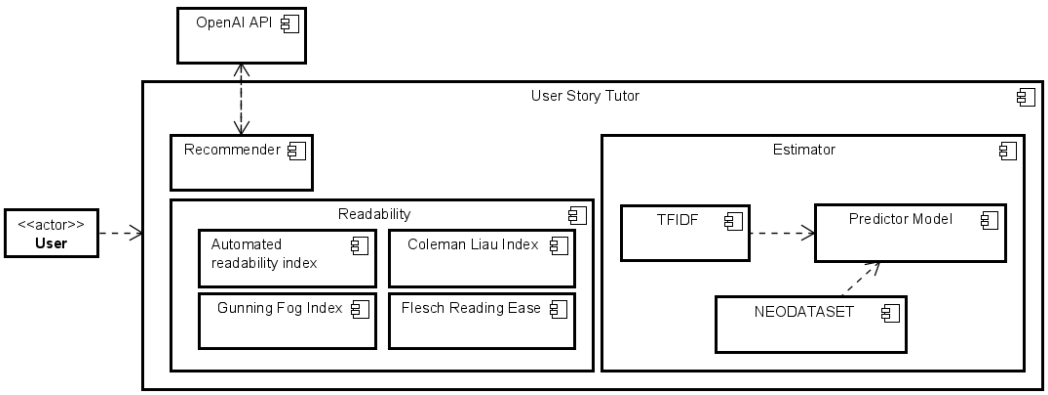
6.3.3 Arquitetura

A ideia é que o UST apoie equipes ágeis durante a construção de *User Stories* e auxilie o processo de desenvolvimento durante a fase de preparação de *User Story* e na estimativa de esforço da tarefa em Story Point. O UST consiste em uma aplicação web que pode ser acessada por meio de um navegador em dispositivos móveis, PCs e notebooks. A UST utiliza um LLM fornecido pela OpenIA para recomendar melhorias e apresentar índices de legibilidade. As principais partes da ferramenta são discutidas abaixo.

A aplicação foi desenhada em torno de 3 módulos com funções bem definidas. O módulo Recomendador responde às solicitações de recomendação de *User Stories*. Ele é responsável por manter o prompt e combiná-lo com o texto da nova *User Story*, consultar OpenAI via API e preparar o retorno para apresentação ao usuário. O módulo que realiza estimativas de *User Story* em *Story Points* utiliza um

modelo preditivo, já treinado com dados históricos para auxiliar os desenvolvedores em suas estimativas, servindo de referência para a equipe responsável pela estimativa de esforço. Os índices de legibilidade são extraídos do texto com técnicas básicas de processamento de linguagem natural. Uma imagem da arquitetura é apresentada na Figura 6.3.

Figura 6.3: UST Architecture



Interface

A primeira tela do UST (Figura 6.4) é onde um desenvolvedor da equipe (usuário do UST) informa sua User Story. Qualquer *User Story* de qualquer projeto real pode ser usada. Então, após o desenvolvedor inserir a descrição da *User Story* em formato de texto, ele clica no botão “Analisar”. O UST então inicializa as threads necessárias que acionam as respostas dos módulos existentes.

Recomendador

O módulo de recomendação é um módulo que retorna sugestões para o texto de descrição da *User Story* do desenvolvedor digitado pelo usuário (Figura 6.5). Foi utilizado o modelo fornecido pela OpenAI, especificamente o gpt-3.5-turbo. Os parâmetros usados pelo módulo recomendador são enviados por um prompt oculto apresentado na Tabela 6.1. Eles seguiram um processo de refinamento semelhante à criação de uma chave de busca em uma revisão sistemática da literatura, sendo revisados e adaptados até chegar à versão final apresentada com o apoio do conjunto de dados produzido (NeoDataset).

Figura 6.4: Home Screen

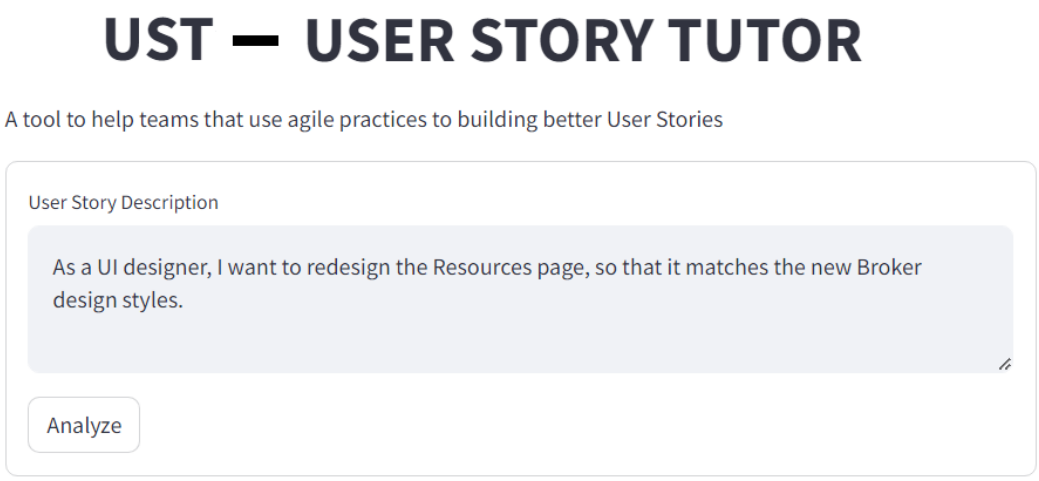


Tabela 6.1: prompt personalizado

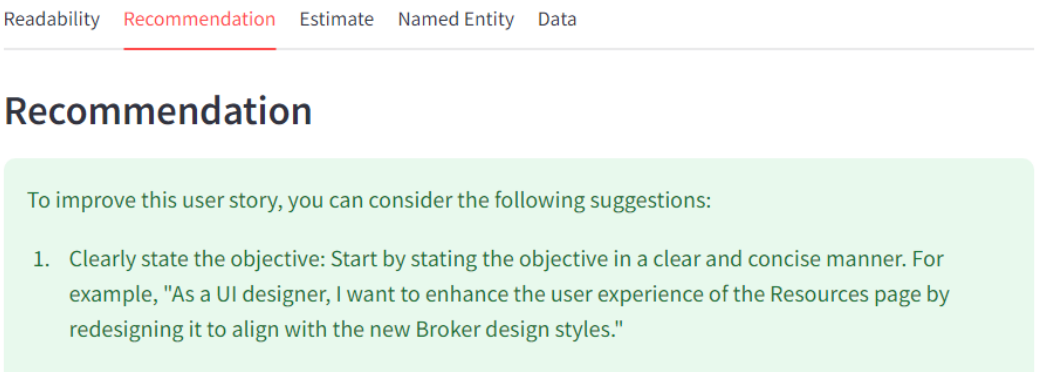
papel	conteúdo
system	You are a scrum master, skilled in creating better *User Stories* for agile software projects
user	How can I improve this user story: + *User Story text informed by user*

Por exemplo, para o módulo Recomendador, uma entrada válida do usuário seria esta User Story: *"Como designer de UI, quero redesenhar a página Recursos, para que ela corresponda aos novos estilos de design do Broker."*. Um possível texto de recomendação que a UST retornaria ao usuário é: *"Para melhorar esta User Story, você pode considerar adicionar mais detalhes e tornar o objetivo mais claro. Aqui está uma versão melhorada da User Story: Como designer de UI, quero redesenhar a página Recursos para melhorar a experiência do usuário e alinhá-la com os novos estilos de design do Broker, aprimorando a consistência geral e o apelo visual do aplicativo. Além disso, você pode refinar ainda mais a User Story especificando as alterações ou melhorias específicas que planeja fazer na página Recursos"*.

Preditor

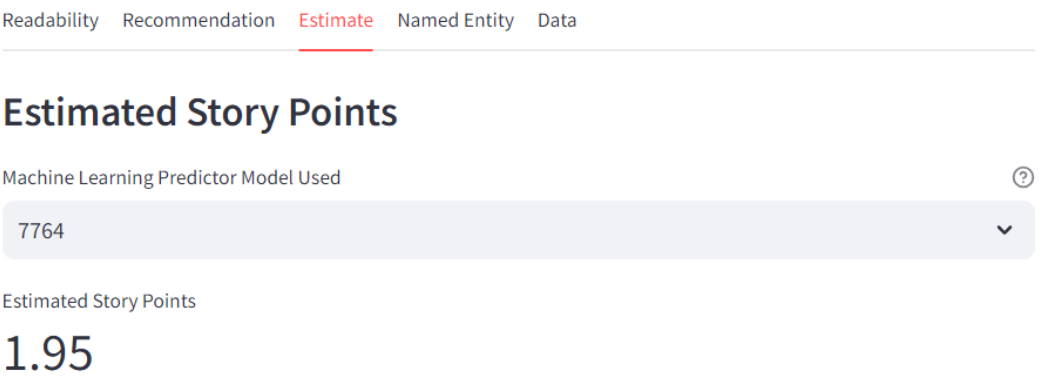
O módulo Preditor (Figura 6.6) realiza uma estimativa de esforço em *Story Points* com base na descrição da User Story. O modelo preditivo e o vetorizador usados são carregados com a biblioteca Joblib. O algoritmo selecionado foi o SV. O texto

Figura 6.5: Exemplo do módulo de recomendação



da *User Story* é transformado em um conjunto de palavras usando a técnica TF-IDF. Em produção, tanto o vetorizador quanto o modelo são carregados na memória para previsão. Após o texto da *User Story* ser transformado em uma matriz com o vetorizador, ele é repassado ao preditor do modelo, que retorna a estimativa em Story Points. O modelo carregado em produção foi previamente treinado com dados do NeoDataset.

Figura 6.6: Módulo Preditor



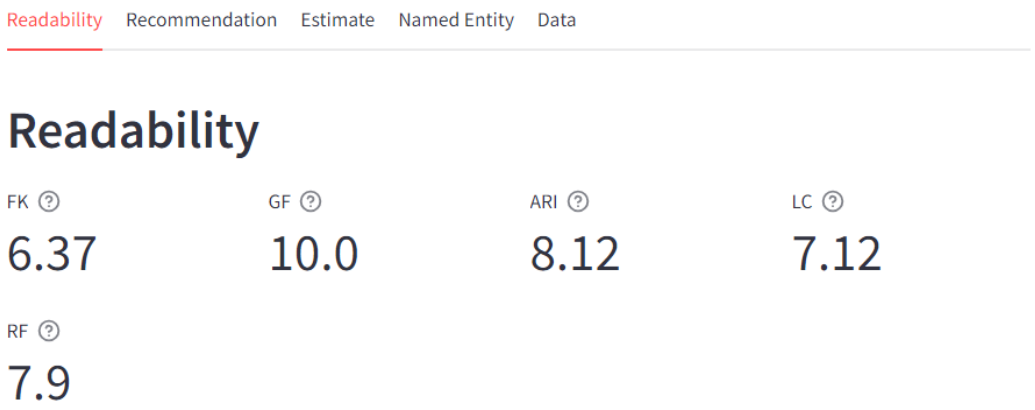
Legibilidade

Para o módulo de legibilidade da *User Story* (Figura 6.7), os índices de legibilidade do texto são extraídos usando a biblioteca textdescriptive (disponível em Python) [1] e apresentados na tela com o componente “metrics” do Streamlit na primeira aba. O objetivo do módulo de legibilidade é permitir que o criador da *User*

[1] https://pypi.org/project/textdescriptives

Story veja alguma medida quantitativa de quão fácil é a leitura do texto de sua User Story.

Figura 6.7: Módulo de legibilidade



Tecnologias

Para codificação, foi utilizado o StreamLit, uma biblioteca para construção de aplicativos de código aberto para aprendizado de máquina e ciência de dados [140]. Como linguagem, foi utilizada Python, uma linguagem de programação que vem aumentando sua participação no mercado, principalmente em aplicações que utilizam aprendizado de máquina [114]. O Módulo Recomendador realiza uma consulta ao OpenAI. As bibliotecas scikit-learn também foram utilizadas [129]. Todo o código-fonte do projeto está disponível no GitHub na seção disponibilidade de artefatos. O UST pode ser testado no StreamLit Cloud, link na seção disponibilidade de artefatos.

6.4 Resultados

Esta seção apresenta uma avaliação qualitativa e quantitativa da ferramenta com o apoio do framework TAM e AttrakDiff e discute os resultados. A pesquisa foi realizada em dezembro de 2023 por meio de um questionário online no Google Forms. O questionário foi primeiramente examinado quanto à abrangência, qualidade e adequação à investigação em questão por um painel de 6 especialistas que

tinham entre eles 7 anos de experiência em desenvolvimento ágil. O questionário utilizou uma escala Likert de 5 níveis para avaliar a concordância do respondente (de “nenhum” ou nível 1), passando por pouco (2), neutro (3), algum (4) até total (nível 5) com afirmações feitas sobre UST. Os comentários e sugestões dos especialistas levaram ao ajuste do questionário, que foi então aplicado a uma amostra de entrevistados.

6.4.1 Caracterização da amostra

A amostra de respondentes da pesquisa é composta por 42 participantes brasileiros. 70% dos que responderam à pesquisa trabalhavam diretamente com metodologias ágeis. Mais da metade deles (56%) havia trabalhado em uma fábrica de software e já atuou como membro de uma equipe de desenvolvimento de software. Destes, 20% são Scrum Masters.

6.4.2 TAM

Cada um dos 4 construtos do TAM é analisado a seguir: percepção de usabilidade, percepção de facilidade de uso, variáveis externas e atitude.

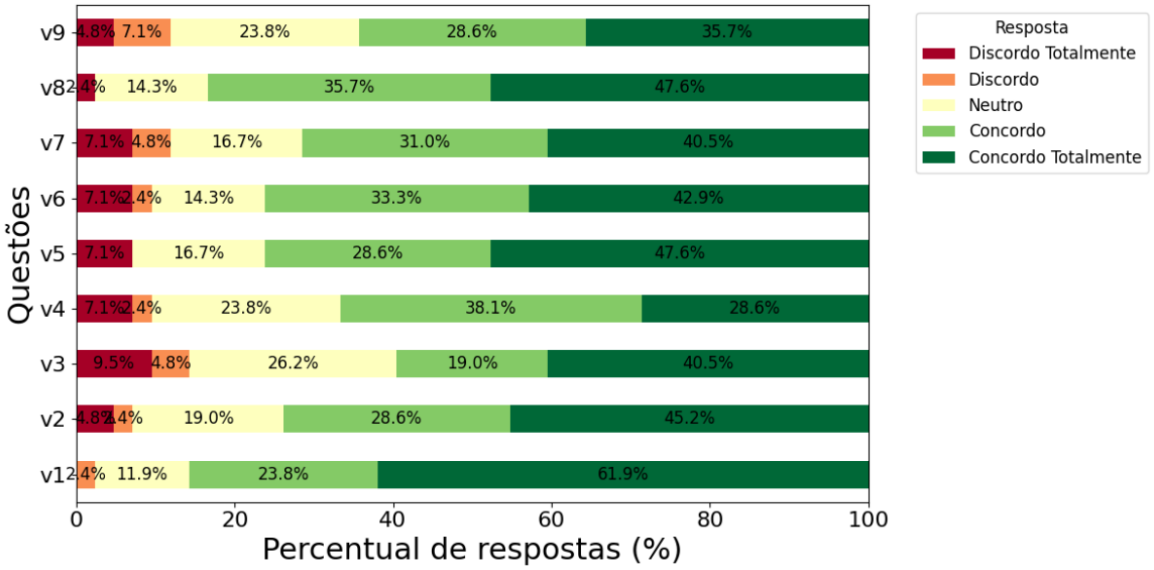
A percepção de usabilidade é o nível em que uma pessoa acredita que utilizar a UST melhora o desempenho de suas tarefas. Para analisar a percepção de usabilidade (Tabela 6.2 e Figura 6.8) foi analisada a distribuição das respostas da escala Likert. OS dados possivelmente indicam que os participantes têm uma atitude positiva em relação à percepção do uso da ferramenta [33].

Avaliando a distribuição das respostas, é possível inferir que os participantes geralmente têm uma atitude positiva em relação à usabilidade percebida da ferramenta (Tabela 6.2 e Figura 6.8).

Tabela 6.2: Percepção de usabilidade

ID	Definição
V1	Usar a ferramenta é útil para melhorar minhas *User Stories*
V2	Aprendi a construir melhores *User Stories* depois de usar a ferramenta

Figura 6.8: Distribuição das respostas (Escala Likert)



A percepção de facilidade de uso é o nível em que a pessoa expressa sua percepção da ferramenta em termos de facilidade de aprendizado e operação. A tabela 6.3 apresenta as perguntas relacionadas à facilidade de uso percebida. Analisando os dados (Figura 6.8) no que diz respeito à boa percepção e facilidade de uso do UST, observamos uma boa aceitação da percepção percebida.

Tabela 6.3: Percepção de facilidade de uso.

ID	Definição
V3	Aprender a usar a ferramenta foi fácil para mim
V4	A busca por informações nessa ferramenta foi simples
V5	Acessar a ferramenta é simples

Uma análise de variáveis externas que fornece uma melhor compreensão do que influencia a utilidade percebida e a facilidade de uso é apresentada na Tabela 6.4 e Figura 6.8, indicando uma pequena dispersão entre as respostas.

Tabela 6.4: Variáveis Externas

ID	Definição
V6	Os atributos de navegação do aplicativo - menu, ícones, links e botões - são claros e fáceis de encontrar
V7	A ferramenta tem uma boa interface

Os dados caracterizados como Atitude, sendo a Intenção do indivíduo em utilizar

a ferramenta, são apresentados na Tabela 6.5 e na Figura 6.8. Da mesma forma, como acontece com os demais construtos, tem-se valores desejados na distribuição das respostas.

Tabela 6.5: Atitude

ID	Definição
V8	Acredito que é melhor usar a ferramenta para ajudar a criar a *User Story* do que não usá-la
V9	Pretendo usar a ferramenta para criar melhores *User Stories* e planejar melhor minhas tarefas

Para a confirmação estatística, foi utilizado o teste de Cronbach [49] para o questionário em escala Likert, a mesma técnica utilizada por Dantas et al. (2019) [33]. O alfa de Cronbach é uma forma de medir a consistência interna de um questionário ou pesquisa. Ele varia entre 0 e 1, com valores mais elevados indicando que a pesquisa ou o questionário é mais confiável. Uma interpretação do alfa de Cronbach é apresentada na Tabela 6.6.

Tabela 6.6: Consistência interna da Pesquisa

Cronbach Alfa	Consistência interna
$0,9 \leq \alpha$	Excelente
$0,8 \leq \alpha < 0,9$	Bom
$0,7 \leq \alpha < 0,8$	Aceitável
$0,6 \leq \alpha < 0,7$	Questionável
$0,5 \leq \alpha < 0,6$	Pobre
$\alpha < 0,5$	Inaceitável

Um limite adotado nesta pesquisa é o índice alfa de Cronbach maior que 0,7 para as variáveis analisadas. Os valores na Tabela 6.7 levam a crer que a consistência interna dessa investigação é aceitável.

Tabela 6.7: Cronbach dos construtos do TAM.

Construtos	Cronbach
Usability	0.81
Ease of use	0.92
External variables	0.87
Attitude	0.73

6.4.3 AttrakDiff

Na Figura 6.9 apresenta-se o portfólio de resultados do teste AttrakDiff [56]. Este teste apresenta fatores que podem ajudar a avaliar melhor a abordagem, complementando o que o framework TAM apresenta. Foi utilizado o teste curto do tipo AttrakDiff, que apresenta 10 questões aos usuários e infere as métricas relatadas a seguir.

Na Figura 6.9 o eixo vertical da visualização do portfólio exibe a qualidade hedônica (inferior = baixa extensão). O eixo horizontal mostra a qualidade pragmática (esquerda = baixa extensão). Dependendo dos valores das dimensões, o produto ficará em uma ou mais regiões. Quanto maior o retângulo de confiança, menos certeza se pode ter sobre a região à qual ele pertence. Um pequeno retângulo de confiança é uma vantagem porque significa que os resultados da investigação são mais confiáveis e menos coincidentes. O retângulo de confiança mostra se os usuários concordam com a avaliação do produto. Quanto maior o retângulo de confiança, mais instáveis são as classificações de avaliação [56]. Assim, as respostas apontam para um pequeno retângulo de confiança no quadrante superior direito, como uma ferramenta orientada para tarefas.

Os valores médios das dimensões AttrakDiff para o produto avaliado são plotados no diagrama na Figura 6.10 . Nesta apresentação, a qualidade hedônica distingue entre os aspectos de estimulação e identidade. Além disso, é apresentada a classificação de atratividade [56].

Na Figura 6.11 apresenta-se a descrição dos pares de palavras. De particular interesse são os valores extremos. Estes mostram quais características são particularmente críticas ou particularmente bem resolvidas [56]. Melhores resultados são colocados no quadrante positivo, o que pode ser inferido a partir dos resultados consolidados na Figura 6.11. Quase todos os itens avaliados ficaram no quadrante positivo, exceto o par (barato-premium).

Figura 6.9: Portfólio de resultados

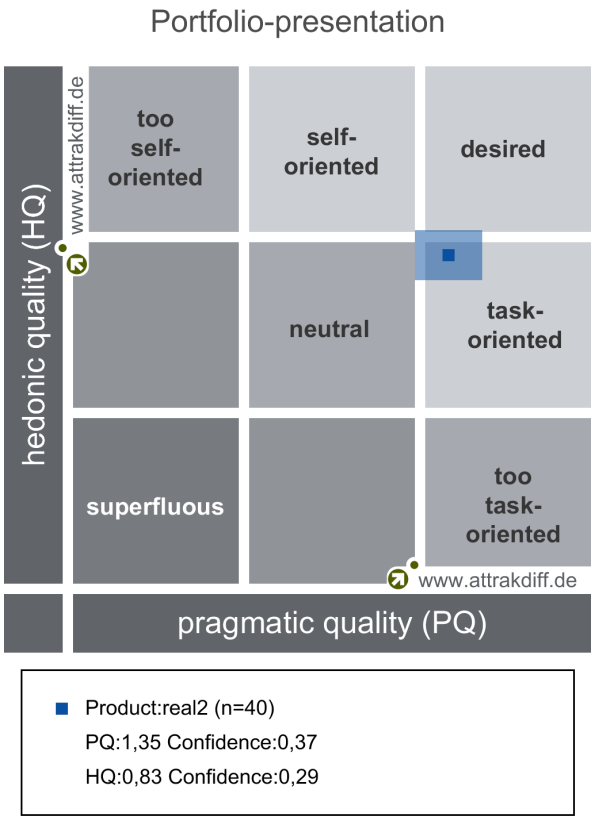


Figura 6.10: Diagrama dos valores médios

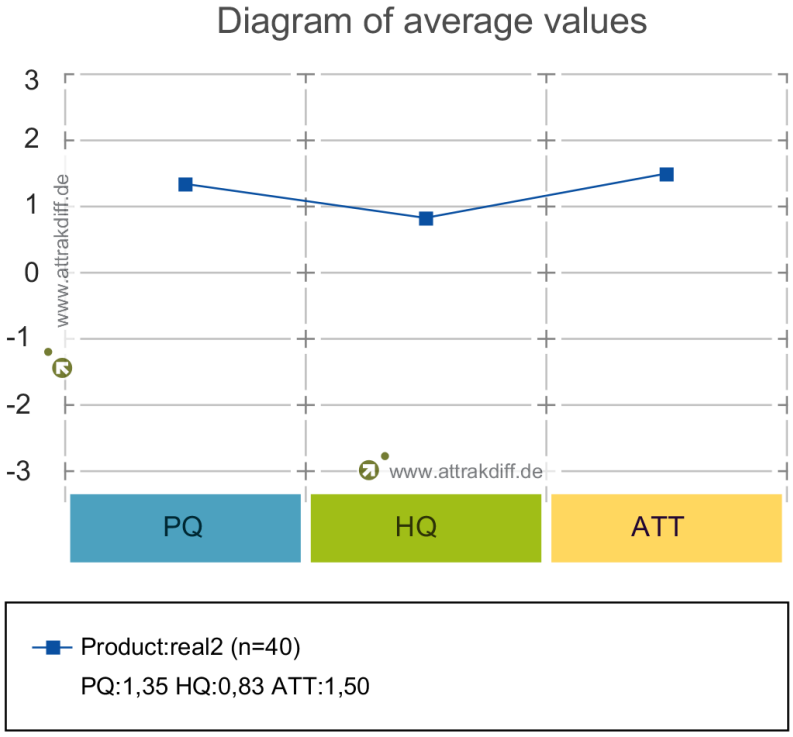
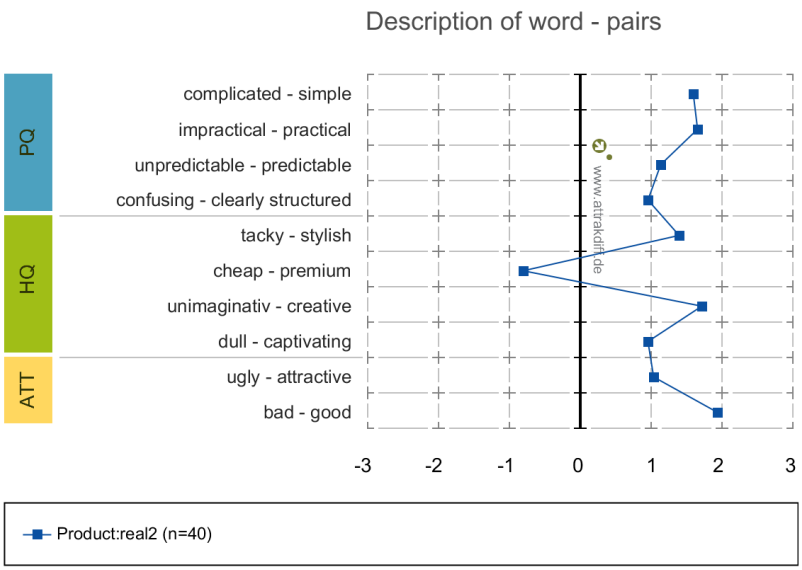


Figura 6.11: Descrição dos pares de palavras



6.5 Trabalhos relacionados

Melhorar a qualidade das *User Stories* é uma linha de pesquisa que vem ganhando força devido aos avanços na inteligência artificial. Geralmente, as abordagens mais clássicas utilizam a transformação para um modelo intermediário, como um caso de uso [40], ou alguma outra técnica de processamento de linguagem natural, com apresentação de relatórios que podem ser interpretados (por exemplo, o AQUSA) [78].

Usar um modelo intermediário para representar a *User Story* adiciona complexidade ao uso da solução, o que pode ser um recurso indesejado. A abordagem adotada não utiliza um modelo intermediário. Ela usa índices de legibilidade de texto, uma técnica já amplamente utilizada para análise de texto em outras áreas (por exemplo, economia), e o uso do LLM da OpenAI com recomendações personalizadas. Além disso, o módulo preditor utiliza aprendizado de máquina com técnicas de processamento de linguagem natural.

O USQA também utiliza técnicas de processamento de linguagem natural para analisar a utilidade, completude e polissemias na criação das *User Stories* [64]. Já o UST, nessa tese traz técnicas adicionais, como recomendação e legibilidade da *User Story* que podem auxiliar ainda mais. A Tabela 6.8 compara a UST às

abordagens da AQUSA e da USQA e ilustra a contribuição da UST em comparação com estes trabalhos.

Tabela 6.8: Comparação com trabalhos relacionados.

Tool	Intermediate Model	Recommend Report	Legibilidade da User Story
UST	Não	Sim	Sim
AQUSA	Não	Sim	Não
USQA	Não	Sim	Não

6.6 Ameaças à validade

Existem algumas críticas na literatura relativas à interpretação numérica de um questionário em escala Likert (por exemplo, no cálculo da média ou mediana de números arbitrários, como na escala Likert) [42]. Para minimizar esse ponto, utilizou-se outro framework para análise de qualidade de software, o AttrakDiff.

A utilização de um modelo LLM disponibilizado pelas empresas via API (por exemplo, o ChatGPT da OpenAI) vincula a solução UST a uma empresa corporativa. Em trabalhos futuros, planeja-se utilizar e validar um modelo de LLM de código aberto.

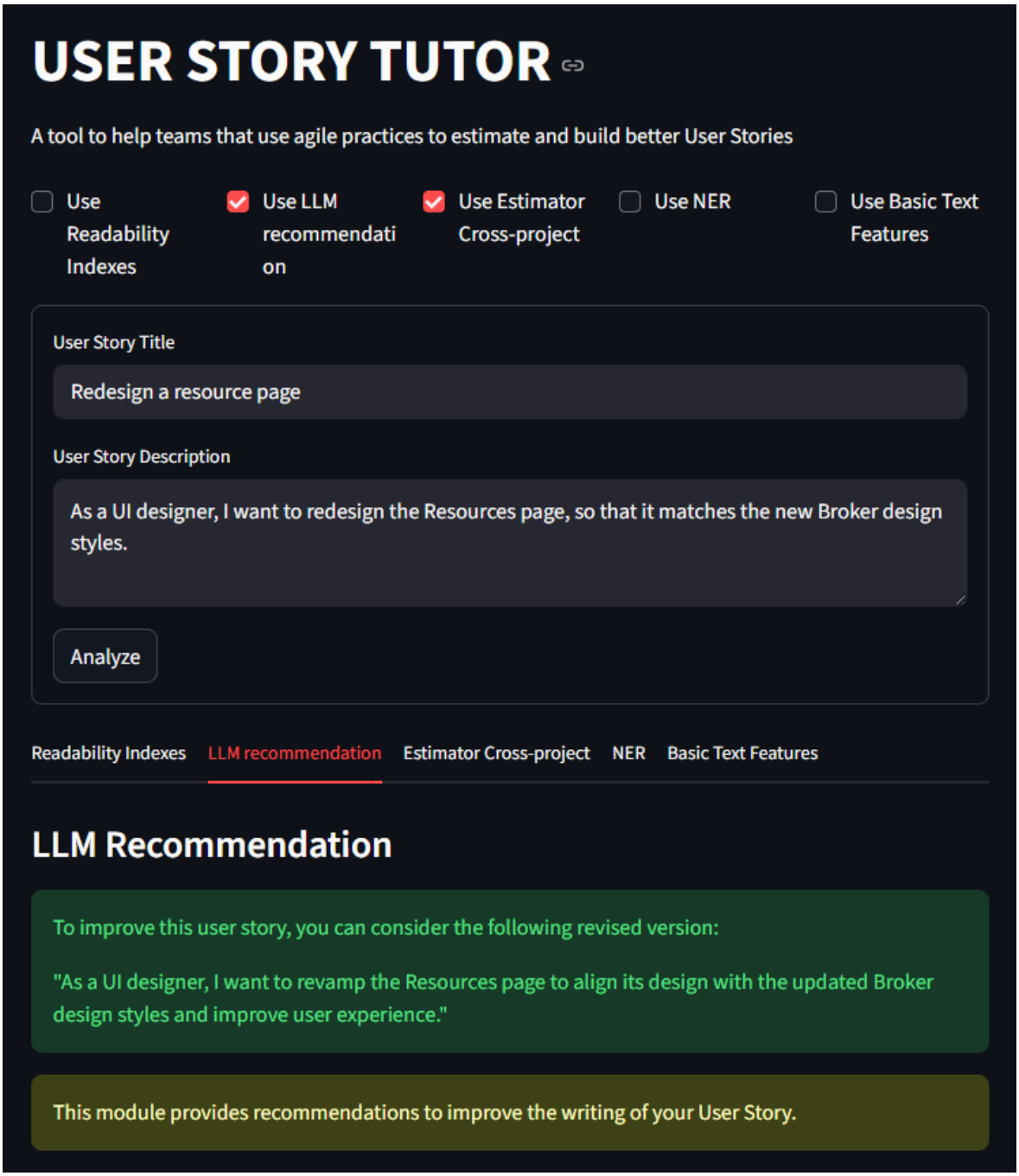
Os índices de legibilidade devem ser interpretados com cautela, pois suas fórmulas utilizam apenas duas variáveis: palavras complexas e frases longas. Portanto, eles não são capazes de medir a coesão e a coerência de uma *User Story*, que abrange fatores semânticos, sintáticos e pragmáticos.

A estimativa em *Story Points* geralmente segue a escala de Fibonacci. Sendo que o preditor retorna um número real entre 0 e 100. Este problema foi tratado como um problema de regressão e não de classificação. Contudo, pode-se obter provavelmente maior interpretabilidade se for utilizada a escala de Fibonacci em vez de números reais.

6.7 Disponibilidade dos artefatos

- Código fonte do projeto `https://github.com/giseldo/userstory`
- Deploy da aplicação `https://userstoryteach.streamlit.app`

Figura 6.12: Tela do *User Story* Tutor, versão final



6.8 Considerações finais

Este capítulo apresentou uma abordagem e avaliação de uma ferramenta para recomendação de boas práticas na escrita de *User Stories* com LLM, além de um módulo de estimativa de *User Stories* com Aprendizagem de máquina e apresentação de índices de legibilidade para a descrição de *User Stories*. A ferramenta foi avaliada por 40 profissionais de engenharia de software. A avaliação foi realizada com os frameworks TAM e AttrakDiff. Os resultados indicam que a UST cumpre os objetivos estabelecidos, com boa aceitação por parte dos usuários pretendidos.

A partir desta investigação, conclui-se que uma ferramenta para auxiliar a construção de *User Stories* é uma técnica viável que, no mínimo, pode ser utilizada para educar as equipes na redação de melhores *User Stories*. Além disso, o UST poderia melhorar a legibilidade do texto da *User Story* através de um “pré-processamento” para a estimativa, fornecendo feedback às equipes de desenvolvimento e possivelmente melhorando a acurácia da estimativa de esforço.

Capítulo 7

Estimativa com Legibilidade e SVR (Neo Legibility Effort Model)

A ciência é muito mais que um corpo de conhecimentos. É uma maneira de pensar.

Carl Sagan

Este capítulo foi publicado na forma de artigo na conferência “ITS2025 - Intelligent Tutoring Systems” com o título “Predictive Model for *Story Points* Leveraging Features Like Readability and Sentiment from *User Story* Description” [96] (disponível no apêndice F).

Este capítulo responde parte da questão de pesquisa “QP 3: Qual técnica pode ser utilizada para uma estimativa que supere os *baselines* identificados (menor erro) para prever *Story Points* a partir do texto das *User Stories*?”. Especificamente a “QP 3.1: Quão bem o modelo com extração de atributos performa em comparação com os baselines?” A motivação foi avaliar a efetividade do modelo, comparando-o com uma técnica simples: a média dos valores dos Story Points.

Resumo

CONTEXTO: Algumas abordagens de AM para estimar os *Story Points* a partir das *User Stories* utilizam TF-IDF com o algoritmo SVM. Porém, essas abordagens de transformação de texto podem não representar bem uma User Story.

HIPÓTESE: A polaridade do sentimento, a subjetividade e a legibilidade da *User Story* são bons preditores da estimativa de esforço.

OBJETIVO: Avaliar um modelo preditivo que utiliza esses atributos.

MÉTODO: Projetaram-se 34 preditores (um para cada projeto considerado), com Support Vector Regression, e comparado seus resultados com a métrica Mean Absolute Error contra dois *baselines* de comparação: a média dos *Story Points* (dos dados de treino) e um modelo que extrai atributos com TF-IDF.

CONCLUSÃO: O modelo *Neo Legibility Effort Model* superou as *baselines* selecionadas em 15 (40%) dos projetos deste conjunto de dados, com menos tempo e recursos, em comparação com 10 da técnica TF-IDF.

7.1 Introdução

Este capítulo detalha o modelo preditivo proposto (apelidado de *Neo Legibility Effort Model*, e em sua forma curta, Neo Legibility Effort Model) para estimar *Story Points* a partir do título e da descrição das *User Stories*.

A hipótese é que a polaridade do sentimento (ou tom) da descrição da *User Story*, a subjetividade da descrição da *User Story* e a legibilidade da descrição da *User Story* são bons preditores da estimativa de esforço em Story Points.

Nas seções seguintes será apresentada a metodologia utilizada, os resultados da comparação com os 2 *baselines* selecionados. As questões que guiaram o estudo e sua motivação são apresentadas na Tabela 7.1.

Tabela 7.1: Questões de pesquisa do estudo.

Questão	**Motivação**
Quão bem o modelo (neo Legibility Effort Model) performa em comparação com a Média dos Story Points?	A motivação foi avaliar a efetividade do modelo comparado-o com uma técnica simples, a média dos valores dos *Story Points* dos dados de treino.
Quão bem o modelo performa em comparação com o *baseline* de uma propostos por outros pesquisadores (TF-IDF)	A motivação foi avaliar a efetividade do modelo comparado-o com uma abordagem já existente e bem estabelecida na literatura.

7.2 Metodologia

A metodologia para a construção do modelo é classificada como explicativa. Os resultados foram analisados por meio de um método quantitativo. Também foram utilizadas as técnicas do Fluxo de Trabalho de AM sugeridas pelo Google (Figura 7.1). Todo o experimento foi implementado na linguagem Python, utilizando a biblioteca scikit-learn, e está disponível para reprodução no GitHub. Parte da análise exploratória para conferência visual está disponível no RPubs e em formato de Slides no GitHub Pages. Veja na seção de disponibilidade de artefatos os endereços.

Figura 7.1: Fluxo de AM do Google

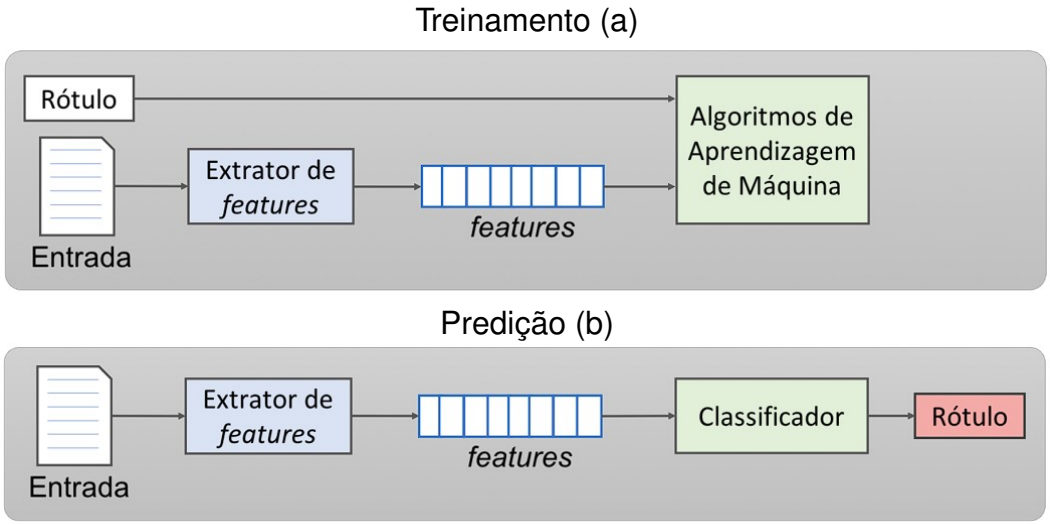
Uma visão macro do processo para a extração dos atributos seguiu o esquema da Figura 7.2. Primeiro, são extraídos atributos do texto. Em seguida, um conjunto de dados formado por esses atributos é utilizado na construção de um modelo preditivo; essa fase é chamada de Treinamento (a). Depois, o modelo é publicado em produção, para ser utilizado em uma predição (b); note que as técnicas de extração de atributos também são necessárias para a predição.

Para extrair as métricas, foi utilizada uma separação entre treino e teste do conjunto de dados (Figura 7.3). Semelhante ao que acontece em outros experimentos, tais como GPT2SP [46], TAWOS [151] e CHOE [23]. Foi utilizado o erro médio absoluto (Mean Absolute Error - MAE) para comparação.

7.2.1 Fluxo de AM

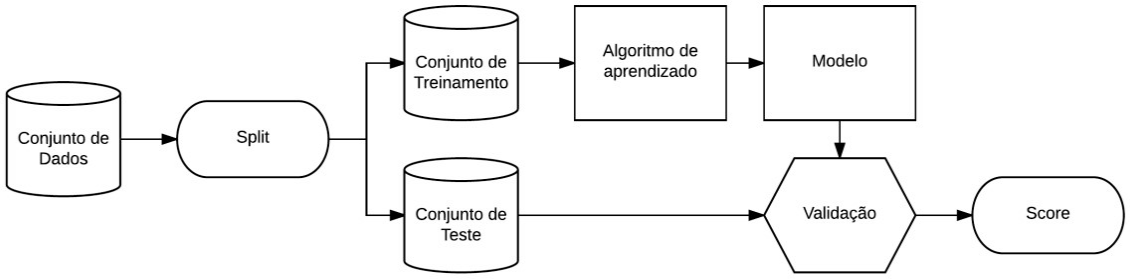
As fases do ciclo de AM (Figura 7.1) utilizadas foram: coleta de dados, exploração dos dados, preparação dos dados, criação, treinamento e validação, e implantação dos modelos. Elas estão detalhadas nas seções seguintes.

Figura 7.2: Descrição da extração dos atributos.



Adaptado de [6]

Figura 7.3: Separação entre treino e teste.



Fonte: Adaptado de [123]

7.2.2 Coleta de dados

O conjunto de dados utilizado foi o NeoDataset. Ele é formado por 34 projetos ágeis, organizados em 34 arquivos CSV. Cada arquivo CSV contém o registro de várias *User Stories* e seus respectivos Story Points, informados pela equipe do projeto.

Foi construído um modelo para cada projeto (no total, 34) e, ao final, foi comparado o Mean Absolute Error de cada um dos projetos. Essa abordagem de um modelo por projeto é nomeada de “modelos individuais”.

7.2.3 Exploração dos dados

Para exemplificação, foi realizada uma análise exploratória com um dos projetos (aleatoriamente selecionado o projeto 7764). Porém, os passos para a análise podem ser replicados para os outros 33 projetos do NeoDataset.

Uma amostra de um conjunto de dados (Projeto 7764) é apresentada na Tabela 7.2. Nela é possível visualizar como os dados se encontravam originalmente. Note que na descrição, é apresentado código HTML, código-fonte, links e outras informações textuais.

As variáveis utilizadas originalmente no conjunto de dados são as colunas: issuekey, created, title, description e o storypoint (Tabela 7.3). Issuekey é um identificador único criado pelo sistema no momento do cadastro de uma Story Point. Created é a data de criação da User Story. Title é um texto curto em uma frase. Description é um texto mais detalhado do que deve ser feito. *Story Point* é a medida de esforço atribuída pelo desenvolvedor.

Primeiro, o título e a descrição das *User Stories* foram unificados em uma só coluna chamada contexto (Figura 7.4). Em seguida, as colunas issuekey, created, title e description foram removidas do modelo.

Remoção outliers

Analisando a contagem dos *Story Points* do projeto 7764, percebe-se que a grande maioria das *User Stories* é medida com poucos Story Points; veja na Fi-

Tabela 7.2: Amostra dos dados do projeto 7764

issuekey	**created**	**title**	**description**	**story points**
29688087	17/01/2020 00:50	Update templates for website merge requests	Relates to &232 and #6109 Goals: * Automatically add relevant details if possible (labels, pings, etc) * Improve clarity of issues * Reduce grooming time * Ensure that all issues contain the [The Five W's] : who, what, when, where, why, and how. * Request [MoSCoW](https://en.wikipedia.org/wiki/MoSCoW_method) information when possible. Implementation Details: - [ ] Update the default MR template. - [ ] Explore the use of checklists in MR templates.	1
29682716	16/01/2020 19:21	Make sure that we Capture Advanced Search in our feature and feature comparison pages	This was raised in the PM & Engineering meeting agenda. > It is under Advanced Global Search https://about.gitlab.com/features/ > Sid: We need it to be added to the feature comparison page, let's check if there are other items missing. > Sid: Which plan is it in? Can't find on => Add here. ## Tasks - [ ] Add elastic search to the feature comparison page - [ ] Check if there is anything else that is missing from the feature comparison page	1

Tabela 7.3: Varriáveis originais do modelo, passo 1

Nome	**Tipo**	**Preditora ou Rótulo**
issuekey	categórica simples	-
created	data	-
title	texto	preditora
description	texto	preditora
storypoints	numérico discreto	rótulo

Tabela 7.4: Varriáveis do modelo, passo 2

Nome	**Tipo**	**Preditora / Rótulo**	Descrição
context	texto	preditora	título + descrição da User Story
storypoint	numérico discreto	rótulo	Story Point informado pelo time daquela USer Story

gura 7.4. Nota-se que somente uma *User Story* tem 128 Story Points; o mesmo vale para uma *User Story* com 32 Story Points. Os dados sugerem a presença de outliers (Figura 7.5). Portanto, foi aplicada uma técnica para a remoção dos outliers, e foram removidos os valores afastados dois desvios padrão acima e abaixo da média. A Figura 7.7 apresenta um boxplot após a remoção dos outliers.

Figura 7.4: Contagem de *Story Points* do Projeto 7764.

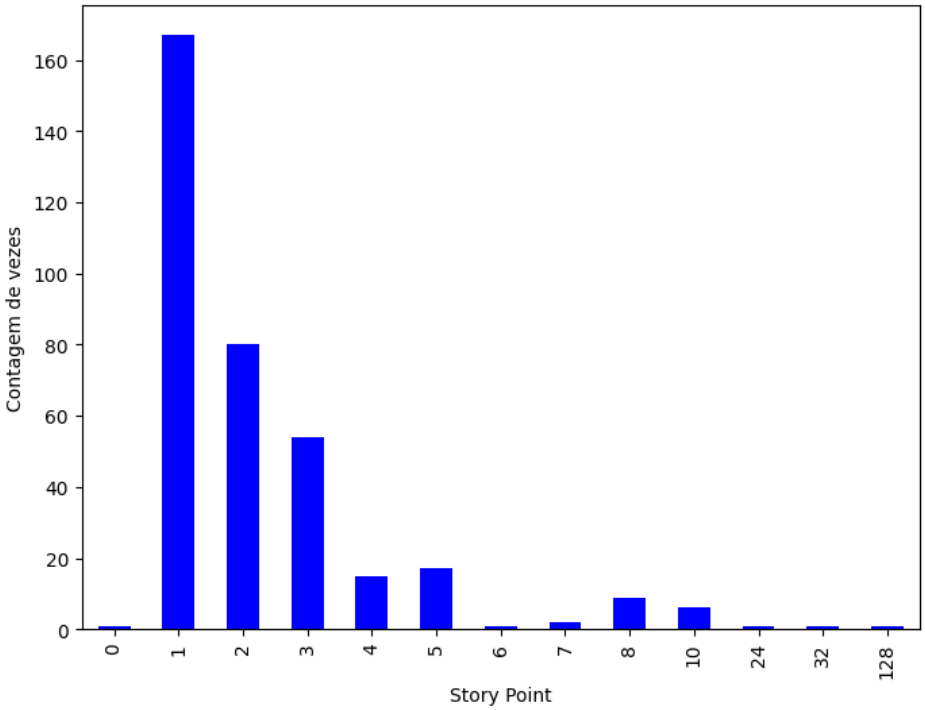


Figura 7.5: Boxplot dos *Story Points* com outliers do Projeto 7764.

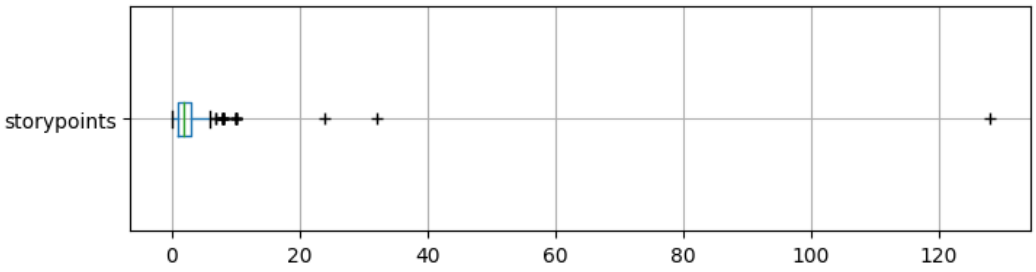


Figura 7.6: Contagem de *Story Points* do Projeto 7764 depois da remoção dos outliers

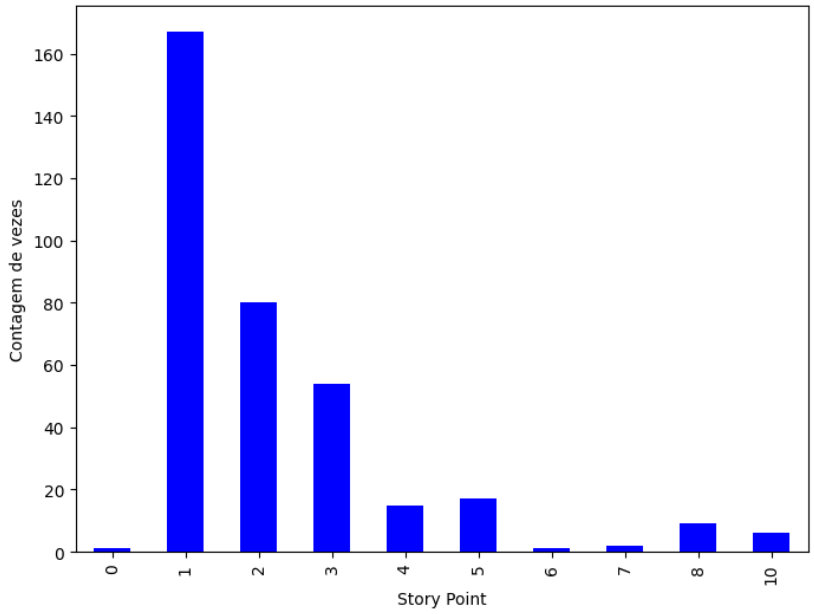
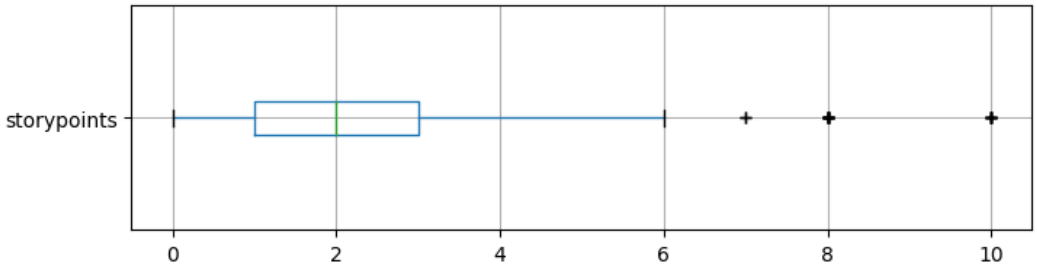


Figura 7.7: Boxplot dos *Story Points* depois da remoção dos outliers do Projeto 7764.



7.2.4 Preparação dos dados

Pré-Processamento

As técnicas de pré-processamento aplicadas são apresentadas na Tabela 7.5. A tabela apresenta o nome da técnica e a expressão regular utilizada. Foram aplicados na coluna contexto os seguintes procedimentos: remover as stopwords, remover os espaços em branco extra, remover as pontuações do texto, remover os números, remover os caracteres especiais e remover as tags HTML. É importante ressaltar que as técnicas de pré-processamento somente foram aplicadas ao modelo do TF-IDF, pois o cálculo das variáveis de legibilidade utiliza a pontuação para calcular o tamanho das frases e a quantidade de palavras; logo, não faz sentido removê-las para o cálculo. Porém, para um modelo que utiliza TF-IDF, a técnica é extremamente importante para diminuir a quantidade de palavras da matriz numérica gerada a partir do texto.

Tabela 7.5: Técnicas de pré-processamento aplicadas na coluna context

Técnica aplicada	**Expressão Regular**
remover as stopwords	Não se aplica
remover os espaços extras em branco	`\s+`
remover as pontuações do texto	`[\w\s]`
remover palavras que contem números	`\b\w*\d\w*\b`
remover os caracteres especiais	`[^A-Za-z0-9\s]`
remover as tags HTML	`<[^>]+>`
remover as URLs e seu conteúdo	`http\S+|www\S+`

Extração dos atributos

Para a extração dos atributos de legibilidade, foi utilizada a biblioteca *textstat* [154], e, para a analise de sentimento da *User Story*, foi utilizada a biblioteca textblob [153].

Os atributos extraídos da coluna *context* foram o Gunning Fog para legibilidade, além de subjetividade e polaridade para o sentimento. A Tabela 7.6 apre-

senta as variáveis do modelo (GUNNING_FOG, POLARITY, SUBJECTIVITY e STORY_POINT) com a tipificação, a sigla e o domínio da variável.

Tabela 7.6: Extração dos atributos do modelo preditivo, passo 3.

Tipo	Nome	Sigla	Domínio	Descrição
Atributo Preditor	GUNNING_FOG	GF	Real	Legibilidade da Story
Atributo Preditor	POLARITY	POL	Real	Sentimento da Story
Atributo Preditor	SUBJECTIVITY	SUB	Real	Subjetividade da Story
Rótulo	STORY_POINT	SP	Real	Story Point da Story

Uma amostra, após a extração dos atributos, é apresentada na Tabela 7.7.

Tabela 7.7: Amostra dos dados do projeto 7764 após extração dos atributos.

gunning_fog	polarity	subjectivity	storypoints
8,85	0,13	0,96	1
7,19	0,11	0,36	1
17,34	0,07	0,37	1
14,98	0,08	0,43	1
11,86	-0,13	0,76	1

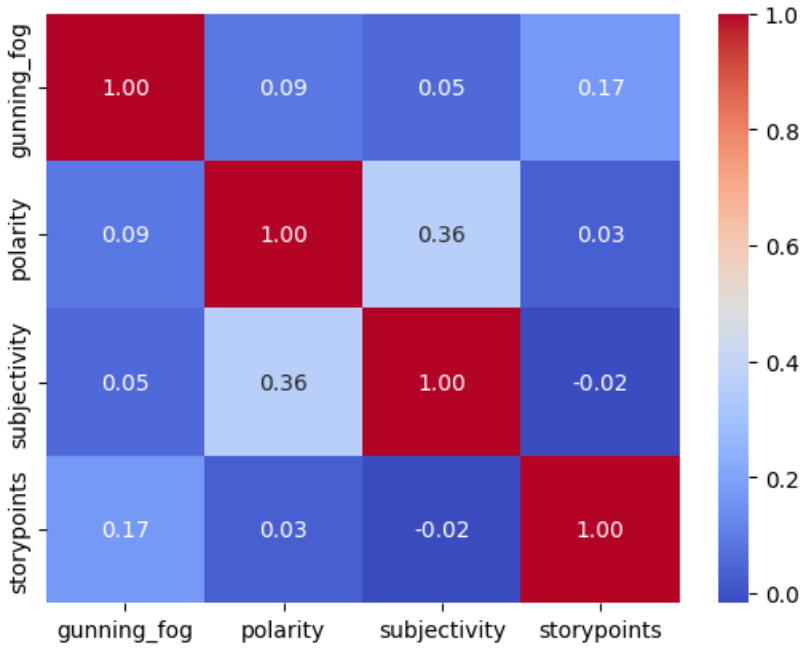
Correlação

Uma correlação entre os atributos é apresentada na Figura 7.8. É possível perceber uma correlação fraca entre o gunning_fog e a quantidade de Story Points.

7.2.5 Criação, treinamento e validação

O algoritmo supervisionado utilizado foi o Support Vector Regressor. Dois *baselines* foram utilizados: a média e outra técnica apresentada anteriormente como solução para o mesmo problema, o TF-IDF [111]. Para a comparação entre os modelos, foi utilizado o erro absoluto médio.

Figura 7.8: Correlação entre os atributos do projeto 7765



Note uma correlação fraca entre gunning_fog e story point.

Extração das métricas para validação

Para o conjunto de treinamento, foram utilizadas 70% das primeiras observações do conjunto de dados, e para o conjunto de teste, os 30% restantes, conforme realizado em pesquisas anteriores [150].

7.3 Resultados

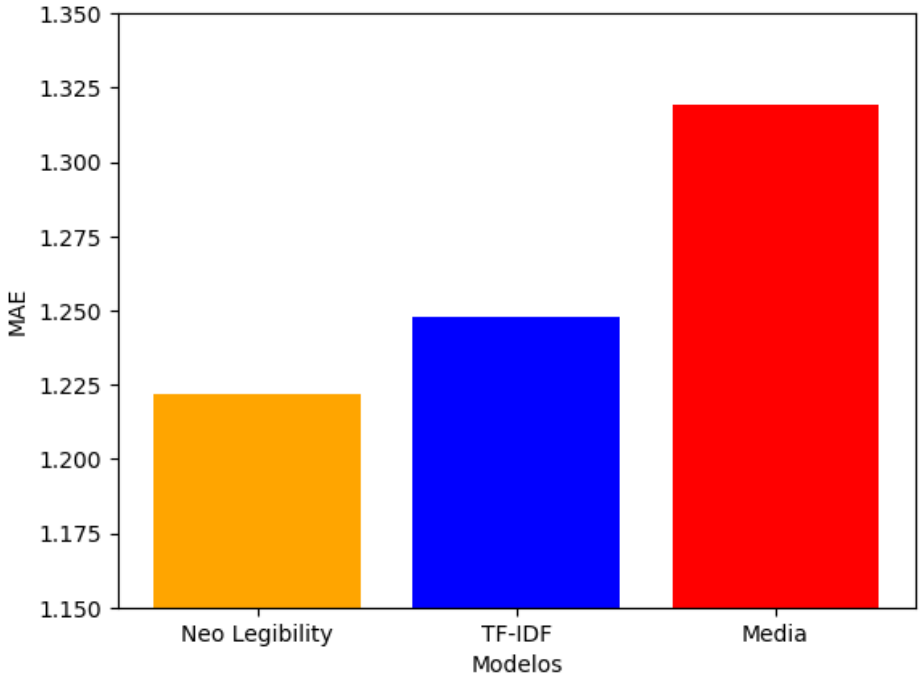
Esta seção apresenta a comparação com um dos projetos (7765) e, em seguida, consolida os resultados dos outros 33 projetos.

7.3.1 Um Projeto

Será utilizado como exemplo um dos projetos, o 7765, escolhido aleatoriamente. O resultado do MAE do modelo 7765 é apresentado na Figura 7.9. O MAE do modelo proposto (Neo Legibility Effort Model) foi menor do que o MAE dos dois *baselines* selecionados: o TF-IDF com SVR e a Média dos *Story Points* dos dados

de Treino. Ou seja, para este projeto, a técnica apresentou um MAE menor. O que confirma a hipótese deste projeto.

Figura 7.9: Comparação do MAE entre os modelos para o projeto 7765



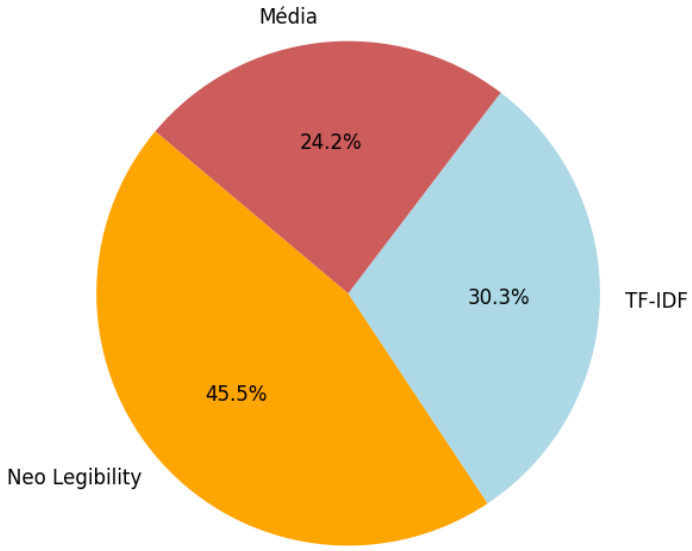
Nota: Quanto menor o MAE melhor.

7.3.2 Todos os Projetos

Em seguida, foram gerados modelos para todos os outros 33 projetos. Uma contagem de quando o modelo teve um MAE menor foi consolidada na Figura 7.10. Logo, o modelo proposto (Neo Legibility Effort Model) teve o menor MAE na maioria dos projetos, em 15 projetos (45%). O modelo TF-IDF com SVR teve um MAE menor em 10 projetos (30%).

Conclui-se que um preditor que utiliza a polaridade do sentimento, a subjetividade e a legibilidade da *User Story* é uma boa alternativa para a predição automática das Story Points, pois apresentou um desempenho superior na maioria dos projetos analisados.

Figura 7.10: Contagem do Melhor modelo para os 34 Projetos.



7.4 Ameaças à validade

O conjunto de dados utilizado (NeoDataset) pode não ser representativo. Sugere-se a confirmação dos achados com outros conjuntos de dados, por exemplo, o TAWOSDataset [146].

Existem indícios de que estimar *User Stories* é uma tarefa que depende de mais informações do que as disponíveis no título e na descrição da *User Story* [130]. Portanto, mais estudos são necessários para confirmar esses achados. Para a adesão a práticas de Open Science e facilitar a reprodutibilidade dos experimentos por outros pesquisadores, os artefatos foram disponibilizados no GitHub.

7.5 Disponibilidade de artefatos

- O código fonte em Python está disponível para reprodução no GitHub `https://github.com/giseldo/Neo_Legibility_Python`.
- Parte da análise exploratória para conferência visual está disponível no RPubs `https://rpubs.com/giseldo/storypoint` e em formato de Slides no GitHub Pages `https://giseldo.github.io/Neo_Legibility_R_Python/`

7.6 Considerações finais

O modelo preditivo *Neo Legibility Effort Model* superou as *baselines* em 15 dos 34 projetos deste conjunto de dados, apresentando uma quantidade significativamente menor de tempo de execução.

O modelo *Neo Legibility Effort*, que utiliza indicadores de legibilidade como características extraídas do texto para estimar Story Points, apresentou um menor MAE e um tempo de execução reduzido em comparação com os baselines.

Ele pode ser utilizado em sistemas automatizados de estimativa de esforço, como o realizado no *User Story* Tutor (descrito no capítulo 6).

Entretanto, existem indícios de que os atributos extraídos do texto não representam toda a semântica necessária para descrever as *User Stories*, apesar do erro reduzido. O capítulo 8 a seguir sugere uma abordagem para este problema, que é a representação semântica do texto da user story, visando a estimativa automática de *Story Points* com o uso de LLM.

Capítulo 8

Estimativa com LLM ajustado (Neo LLM Predictor)

A “inteligência” desses sistemas é uma miragem — o que se vê não é compreensão, mas um ajuste estatístico de padrões.

Emily Bender

Artigo publicado na conferência CBSOFT 2025, Congresso Brasileiro de Software (CBSoft), na trilha “SBES 2025 - XXXIX Simpósio Brasileiro de Engenharia de Software” [97].

Este capítulo aborda parte da questão de pesquisa “QP 3: Qual técnica pode ser utilizada para uma estimativa que supere os *baselines* identificados (menor erro) para prever *Story Points* a partir do texto das *User Stories*?”. Especificamente, a questão “QP 3.3: O modelo com LLM tem menor erro em comparação com os baselines?” busca avaliar se há melhorias na estimativa ao se utilizar LLM. Foi criado, portanto, um modelo LLM ajustado, que está disponível no Hugging Face (chamado de Neo LLM Predictor e utilizado na aplicação Neo Estimator V2 - Apêndice G.

Resumo

CONTEXTO: A estimativa de esforço em projetos ágeis continua sendo um desafio recorrente, especialmente quando os *Story Points* precisam ser inferidos ape-

nas a partir do texto das *User Stories*. Estudos anteriores, inclusive no Capítulo 7 desta tese, focaram principalmente em abordagens de aprendizagem de máquina para prever o esforço. No entanto, a recente disponibilidade de Modelos de Linguagem de Grande Escala (LLMs) oferece uma alternativa viável.

OBJETIVO: O objetivo do capítulo é investigar a eficácia dos LLMs na estimativa de story points. Um derivado do modelo BERT foi ajustado (fine-tuning) e comparado, em relação ao erro médio absoluto, a três baselines: (i) um modelo preditivo tradicional baseado em vetores TF-IDF acoplados a um classificador de Regressão Linear; (ii) um modelo LLM em zero shot; e (iii) um modelo LLM em few shot. Foi utilizado um conjunto de dados de *User Stories* de projetos reais de desenvolvimento de software ágil, o Deep-SE, que contém várias *User Stories* de 16 projetos open-source diferentes, retirados do Jira [9].

RESULTADOS: Os resultados sugerem que o LLM ajustado apresentou um MAE menor na maioria dos projetos. Os achados sugerem que, apesar do maior custo computacional, os LLMs constituem uma alternativa com menor erro para a estimativa de esforço em comparação com as técnicas analisadas.

8.1 Introdução

Este capítulo tem como objetivo avaliar modelos LLM (zero-shot, few-shot e fine-tuned) para a estimativa do esforço em Story Points. Contrastando seu desempenho preditivo com a abordagem clássica de Regressão Linear e a extração de características pela técnica de frequência inversa de documentos (TF-IDF) [92]. A hipótese é que um modelo de LLM ajustado (fine-tuning) apresenta um erro médio absoluto (MAE) inferior ao dos *baselines* selecionados.

O modelo utilizado foi o distilbert-base-uncased, que oferece um bom equilíbrio entre qualidade semântica, eficiência computacional e facilidade de ajuste para um problema de regressão sobre textos. Esse modelo permite treinar e implantar uma estimativa de esforço, mesmo em hardware modesto e com conjuntos de dados medianos.

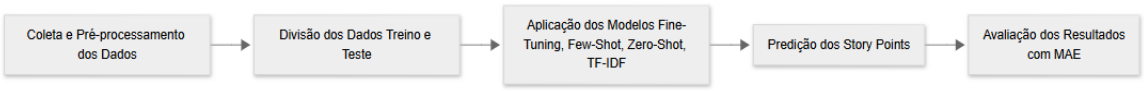
O modelo ajustado neste experimento, denominado de distilbert-

base-uncased-story-point (nomeado no Hugging Face como distilbert_bert_uncased_finetuned_story_point), não perdeu poder de generalização quando foi treinado com dados *Cross-Project*. Esta generalização resultou em resultados concretos: o modelo ajustado superou o LLM few-shot em 14 de 16 projetos e o *baseline* TF-IDF + RL em 11 dos 16 projetos, segundo a métrica MAE.

8.2 Metodologia

A metodologia do estudo foi experimental e aplicada [167]. Foi utilizada a técnica de predição direta. Os modelos preditivos construídos preveem um *Story Point* contínuo que é arredondado para o número da sequência de Fibonacci [144] mais próximo. Os procedimentos realizados em alto nível estão apresentados na Figura 8.1.

Figura 8.1: Procedimentos realizados



Conjunto de dados: Foi conduzido um experimento utilizando o conjunto de dados Deep-SE [23]. O conjunto de dados utilizado contém 23.313 *User Stories* de 16 projetos diferentes e apresenta 5 colunas. O conjunto de dados foi consolidado em um arquivo CSV e disponibilizado no Hugging Face (`https://huggingface.co/datasets/giseldo/deep-se`) para facilitar o acesso e a reprodutibilidade dos experimentos. Os dados incluem o nome do projeto, o identificador (ID), a descrição da *User Story*, o título da *User Story* e os story points. Um exemplo das primeiras colunas do conjunto de dados é apresentado na Tabela 8.1.

O conjunto de dados utilizado foi consolidado por Choetkiertikul et al. [23]. Foi disponibilizado um link `https://github.com/SEAnalytics/DSL_reading_list` para o dataset; no entanto, ao acessar o link em maio de 2025, o conjunto de dados não estava disponível para download. Entretanto, Tawosi et al. [149] reproduziram o experimento de Choetkiertikul com o conjunto de dados Deep-SE e disponibilizaram novamente esse conjunto de dados no link `https:`

Tabela 8.1: Amostra do conjunto de dados

Projeto	Chave	Título	Descrição	Story Point
appceleratorstudio	TISTUD-6	Add CA against object literals in function...	The idea here is that if our met...	1
appceleratorstudio	TISTUD-9	Update branding for Appcelerator plugin...	At least fix feature icons, asso...	1
appceleratorstudio	TISTUD-11	Create new JSON schema for SDK team	Create JSON schema containing pr...	1
appceleratorstudio	TISTUD-13	Create Project References Property Page	Create property page for project...	1
appceleratorstudio	TISTUD-16	New Desktop Project Wizard	Desktop (need to convert existin...	1

`//figshare.com/s/709c7e18c52e4264b70e`. Estes foram os arquivos que foram utilizados, consolidados e disponibilizados no Hugging Face.

Esses dados do Deep-SE foram coletados de 16 projetos de código aberto extraídos do Jira [23] . Quanto ao domínio e às empresas, os 16 projetos provêm de 9 diferentes repositórios ou organizações de código aberto. Os projetos do conjunto de dados não são da mesma empresa, mas sim de diversas entidades. Os projetos demonstram diversidade em termos de domínios de aplicação e características técnicas, incluindo diferentes linguagens de programação. Todas as *User Stories* foram criadas e registradas no sistema de rastreamento de problemas Jira [9]. O Jira é um dos sistemas amplamente utilizados que suporta o desenvolvimento ágil, incluindo a estimativa de Story Points.

Técnicas: Foram utilizadas as técnicas zero-shot, few-shot e fine-tuning. Para o fine-tuning, foi criado um único modelo *Cross-Project* . Para os outros casos, foram criados 16 modelos, um para cada projeto do conjunto de dados. O *Fine-Tuning* do modelo LLM foi realizado com o modelo BERT, especificamente o distilbert-base-uncased [125]. Foi utilizado o Google Colab fornecido pelo Google para o *Fine-Tuning* do modelo. O *Fine-Tuning* do modelo durou poucos minutos em um

processador A100. Para a extração das métricas com o modelo zero-shot e few-shot, foi utilizado o modelo Gemma 3:4b em um PC com Processador Ryzen 5 ou Core i5, com 8 GB de memória RAM e 250 GB de HD ou SSD.

Foi escolhido o modelo Gemma 3:4b por sua natureza *open weight*, leveza computacional e desempenho competitivo em tarefas de compreensão de linguagem natural. Esse modelo foi projetado para rodar localmente com baixo consumo de memória, sendo compatível com ambientes de execução otimizados, permitindo a execução eficiente mesmo em computadores pessoais. Essa característica é essencial para viabilizar experimentos com LLMs sem dependência de GPUs de alto desempenho ou serviços em nuvem. Além disso, o Gemma3:4b oferece suporte nativo para abordagens few-shot e zero-shot, permitindo a realização de inferências com base em poucos exemplos ou apenas instruções em linguagem natural, sem a necessidade de reconfiguração dos pesos do modelo [118, 115]. A inclusão desse modelo no experimento visa explorar alternativas acessíveis e reprodutíveis para a estimativa de esforço baseada em *user stories*, complementando a avaliação de modelos ajustados via *fine-tuning* , como o *distilbert-base-uncased* [125]. Dessa forma, é possível comparar diferentes paradigmas de aplicação de LLMs no contexto de estimativas ágeis com base textual, ampliando o entendimento sobre custo-benefício e desempenho em diferentes cenários computacionais.

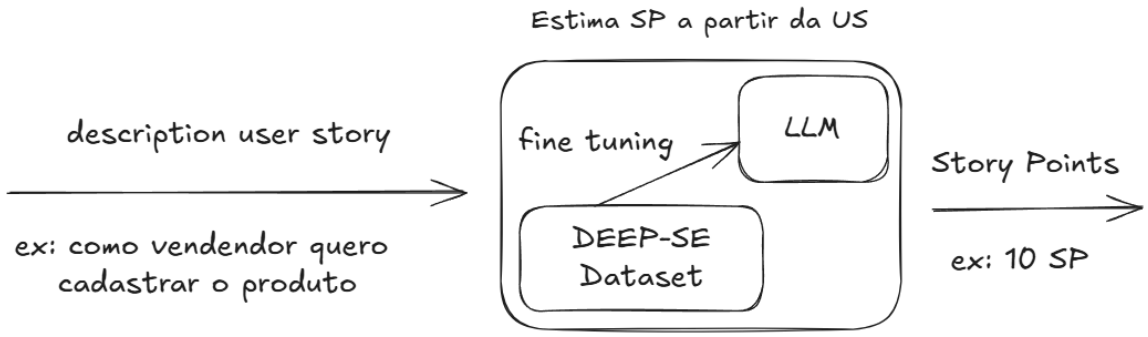
Pré-processamento: Para o pré-processamento, foram utilizadas as técnicas de remoção das tags HTML e a remoção das URLs, além da tokenização.

Treino e teste: Em todos os casos, a separação do conjunto de dados e do conjunto de treino foi de 70% para treino e 30% para teste (geração das métricas). Foram utilizados os primeiros 70% do conjunto de dados estratificados por projetos (nota: o conjunto de dados estava ordenado por projeto e por data de inclusão da *User Story*, ou seja, os 70% primeiros registros de cada projeto estão ordenados de forma temporal - da mais antiga para a mais recente). O modelo ajustado (para *Fine-Tuning* e para todos os outros) só conhece os 70%.

Ou seja, os 30% utilizados para a geração das métricas não eram conhecidos por nenhum dos modelos criados.

Prompt: Os prompts utilizados (Prompt 8.1 e Prompt 8.2) foram construídos

Figura 8.2: Arquitetura alto nível



com base em boas práticas, tais como a contextualização da tarefa, o uso de exemplos explícitos e o reforço da instrução por redundância. Foram realizados testes preliminares empíricos com variações da redação do prompt, até que se chegou a uma versão que maximizasse a clareza das instruções e a consistência das respostas.

8.2.1 Distilbert-base-uncased

O modelo distilbert-base-uncased [3] foi a escolha para estimar o esforço a partir das *User Stories* por diversas razões. Ele foi lançado em 2019, sendo menor e mais rápido que o BERT e otimizado para tarefas que processam frases ou sentenças. Sua versão uncased é útil para descrições de código e problemas, já que não considera diferenças de maiúsculas/minúsculas. Ele é pré-treinado em um grande corpus, o que ajuda na generalização, e tem uma boa arquitetura para esse tipo de tarefa, além de exigir hardware acessível, com uma configuração mínima típica de uma CPU Ryzen 5 ou Core i5 com 8 GB de memória RAM.

O distilbert resulta de um processo de knowledge distillation que reduz em aproximadamente 40% o número de parâmetros e acelera a inferência em cerca de 60%, preservando 95–97% da acurácia do BERT-base (original) em benchmarks de compreensão de linguagem natural [125]. Essa capacidade de fornecer representações ricas com custo computacional inferior é particularmente vantajosa em ambientes de hardware moderado. Com apenas 67M de parâmetros [125], a vari-

ante uncased cabe em uma GPU modesta (por exemplo, 6 GB de RAM da placa de vídeo), possibilitando *Fine-Tuning* e inferência dentro de recursos computacionais restritos. Assim, torna-se viável re-treinar o modelo à medida que novos dados do projeto se acumulam, mantendo a acurácia sem investir em infraestruturas onerosas.

Do ponto de vista linguístico, as *User Stories* analisadas possuem extensão média inferior a 225 tokens (quando usado o tokenizador tiktoken com o GPT-2); especificamente, 69% das *User Stories* estão abaixo de 128 tokens (7215 estão acima de 128 tokens e 16098 abaixo). Logo, a maioria das *User Stories* é observada pelo *Transformer* sem truncamento (abaixo de 128 tokens), preservando as dependências de longo alcance.

Por fim, o amplo suporte no ecossistema Hugging Face para os modelos do tipo BERT e para outros modelos simplifica a reprodutibilidade e a integração em pipelines de tarefas de processamento de linguagem natural. Pensando nisso, o modelo gerado (distilbert-base-uncased-story-point) foi disponibilizado no Hugging Face para uso por outros pesquisadores, com o link disponível na seção de disponibilidade dos artefatos.

8.2.2 Pipeline Fine Tuning

O pipeline de treinamento utilizando o modelo distilbert-base-uncased teve o objetivo de prever o valor de *Story Points* a partir da descrição textual de tarefas em projetos ágeis. O processo envolveu a preparação dos dados, tokenização, definição do modelo, configuração de treinamento e avaliação de desempenho.

Primeiramente, o conjunto de dados Deep-SE foi dividido em 70% para o subconjunto de treino e 30% para o teste inicial.

Na etapa de tokenização, foi aplicada a tokenização BERT com truncamento e padding para limitar as sequências a 128 tokens. Para a construção do modelo, utilizou-se um modelo distilbert com uma única saída contínua para prever valores reais (regressão), que depois foram arredondados para a sequência de Fibonacci.

Na configuração do Treinamento, definiu-se o uso de otimização com taxa de

aprendizado, tamanho de batch, número de épocas e estratégias de salvamento e avaliação, respectivamente, 2e-5, 8, 4 e “epoch”.

A métrica utilizada para avaliação interna foi o erro quadrático médio (MSE). Note que a métrica MSE é utilizada internamente na construção do modelo de fine-tuning. A métrica utilizada para comparar com os outros *baselines* foi o erro médio absoluto, dada a facilidade de interpretação do erro médio absoluto em relação ao erro médio quadrático e seu uso na comparação entre modelos neste domínio por outros pesquisadores.

Neste pipeline, foi gerado um único modelo *Cross-Project* que foi utilizado para a extração das métricas, diferente dos outros pipelines que foram criados com 16 preditores, um para cada projeto. Neste pipeline, o conjunto de treinamento foi uma amostra estratificada por projeto.

8.2.3 Pipeline Few Shot

Para um dos modelos do baseline, foi implementada uma abordagem de LLM com few-shot learning. Foi utilizado o modelo Gemma3:4b com o suporte do ollama (software para execução de LLMs em hardware local). Essa abordagem consiste em construir um modelo capaz de prever o número de *Story Points* a partir da descrição textual das *User Stories*, fornecendo, como contexto, exemplos representativos de estimativas anteriores. Foram criados 16 preditores (ou modelos preditivos) e avaliados individualmente com o MAE.

Inicialmente, foi realizado o pré-processamento do conjunto de dados. Foram removidas aquelas *User Stories* com valores nulos ou iguais a zero. O conjunto de dados resultante foi particionado em dois subconjuntos: 70% para treinamento e 30% para teste.

Para configurar o cenário de few-shot learning, foi selecionada uma amostra de 10% do conjunto de treinamento, a qual serviu como base para a construção de exemplos de entrada. Cada exemplo incluía o texto da *User Story*, sua descrição e o valor real de *Story Points* atribuídos. Esses exemplos foram utilizados como contexto em prompts estruturados que instruem o modelo a realizar uma nova esti-

mativa. O prompt utilizado é apresentado na Prompt 8.1. Note que no Prompt 8.1, no final do prompt, há uma redundância para reforçar que o modelo siga a instrução, o que é uma técnica bastante utilizada na criação de prompts: repetir o texto para enfatizar a importância da instrução.

Prompt 8.1: Prompt Few Shot

```
1 Você é um especialista em estimativa de esforço para projetos ágeis. Com
    base no texto da \textit{User Story} e sua descrição, estime os \
    textit{Story Points} necessários para completar a tarefa. Considere
    fatores como complexidade, volume de trabalho e riscos implícitos.
    Retorne apenas um número inteiro representando os \textit{Story Points
    } (ex.: 1, 2, 3, 5, 8, 13, etc.). Aqui estão alguns exemplos de
    estimativas anteriores:
2 <AQUI LOOP COM 10% DAS ESTIMATIVAS>
3     Exemplo:
4     Texto: {'user_story_text'}
5     Descrição: {'description'}
6     Story Points: {'real_story_points'}
7 <FIM LOOP>
8 Agora, estime os \textit{Story Points} para:
9 Texto da user story: {user_story_data['user_story_text']}. Descrição da
    user story: {user_story_data['description']}. Retorne sempre apenas um
     número inteiro representando os \textit{Story Points} (ex.: 1, 2, 3,
    5, 8, 13, etc.). Não retorne nenhum texto além do número inteiro. Não
    retorne nenhum texto além do número inteiro.
```

Em seguida, as tarefas do conjunto de teste foram processadas individualmente. Para cada uma delas, foi construída uma requisição textual ao modelo, incluindo os exemplos anteriores, o título e a descrição da nova tarefa, solicitando como resposta apenas um número inteiro correspondente à estimativa de esforço. O modelo LLM, configurado com baixa variabilidade (baixa temperatura/aleatoriedade: 0,3), retornava uma predição para cada tarefa, que era, então, registrada para posterior análise.

Os resultados foram organizados em uma estrutura tabular contendo identificadores das tarefas, valores reais e valores estimados, viabilizando a análise quantitativa do desempenho preditivo da abordagem. Essa estratégia foi aplicada em múltiplos projetos distintos, com o objetivo de avaliar a robustez e a capacidade de generalização do modelo em diferentes domínios de desenvolvimento ágil de software.

8.2.4 Pipeline Zero-shot

No pipeline zero-shot, que é bem semelhante ao few-shot, também foi utilizado o modelo Gemma3:4b com o suporte do ollama, porém, sem passar dados auxiliares para o LLM realizar a estimativa. No Prompt 8.2 é apresentado o prompt utilizado neste baseline.

Prompt 8.2: Prompt Zero Shot

```
1 Você é um especialista em estimativa de esforço para projetos ágeis. Com
    base no texto da \textit{User Story} e sua descrição, estime os \
    textit{Story Points} necessários para completar a tarefa. Considere
    fatores como complexidade, volume de trabalho e riscos implícitos.
    Retorne apenas um número inteiro representando os \textit{Story Points
    } (ex.: 1, 2, 3, 5, 8, 13, etc.). Texto da user story: {'
    user_story_text'}, Descrição da user story: {'description'}. Retorne
    sempre apenas um número inteiro representando os \textit{Story Points}
     (ex.: 1, 2, 3, 5, 8, 13, etc.). Não retorne nenhum texto além do nú
    mero inteiro. Não retorne nenhum texto além do número inteiro.
```

8.2.5 Pipeline TF-IDF-RL

Neste pipeline, foi implementada uma combinação de Term Frequency–Inverse Document Frequency (TF-IDF) com Regressão Linear [93]. O objetivo foi o mesmo dos outros pipelines: prever os *Story Points* atribuídos a *User Stories* a partir do texto presente nos campos de título e descrição da user story.

Inicialmente, os dados foram extraídos do conjunto de dados Deep-SE. Para cada projeto selecionado, as instâncias com valores nulos ou *Story Points* iguais a zero foram descartadas. A seguir, foi criada uma representação textual unificada denominada contexto, e o mesmo procedimento foi realizado para os outros pipelines, resultando na concatenação do título e da descrição de cada *User Story* em um único atributo textual.

Os dados foram divididos em dois subconjuntos: 70% para treinamento e 30% para teste. A vetorização textual foi realizada com o método TF-IDF, que transforma os textos em uma matriz numérica, ponderando a importância do corpus.

A fase de modelagem envolveu o ajuste de um modelo de Regressão Linear aos dados vetorizados do conjunto de treinamento. As predições geradas sobre

o conjunto de teste foram arredondadas para o inteiro mais próximo, simulando o processo de atribuição prática de story points. Para a avaliação, foi calculado o MAE.

8.3 Resultados

Os resultados indicaram que, embora todos os modelos apresentassem alguma capacidade preditiva, o fine-tuned com *Cross-Project* (ou seja, o treino com todos os projetos do conjunto de dados) superou os demais métodos para a maioria dos projetos, conforme mostrado na Tabela 8.2.

Tabela 8.2: MAE dos Modelos Utilizados (menores valores em negrito)

Model	P1	P2	P3	P4	P5	P6	P7	P8	P9	P10	p11	p12	p13	p14	p15	p16
LLM few shot	5.98	7.07	2.12	5.65	7.36	3.05	4.63	2.64	14.50	6.48	5.50	4.51	5.62	**2.56**	4.05	2.33
LLM *Fine-Tuning*	**1.81**	**3.09**	2.19	**3.79**	**5.42**	2.04	2.38	**1.52**	**8.44**	**2.38**	**3.70**	**1.96**	**3.36**	3.66	**2.64**	2.10
LLM Zero Shot	3.19	3.57	7.00	7.57	7.86	7.59	6.13	6.01	10.06	4.58	4.24	5.23	6.31	7.67	3.81	6.19
TF-IDF LR	6.49	3.89	**1.07**	4.93	30.71	**1.31**	**2.05**	2.30	17.10	2.94	5.06	9.26	5.63	308.49	6.45	**1.34**

O MAE obtido pelo modelo fine-tuned foi significativamente mais baixo na maioria dos projetos, sugerindo um erro menor nas estimativas de conclusão dos projetos. Algumas comparações entre as métricas foram realizadas e são descritas a seguir:

Few Shot x Zero Shot: Quando comparado o modelo few shot com o modelo zero shot, o modelo few shot performou melhor na maioria dos projetos, 10 de 16. Enquanto o modelo zero shot teve MAE melhor (ou seja, menor) em somente 6 de 16 projetos. Veja a Figura 8.3,

Few Shot x Fine-tuning: Quando comparado o modelo Few shot com o modelo Fine-tuning, o modelo *Fine-Tuning* performou melhor na maioria dos projetos, 14 de 16. Enquanto o modelo Few Shot performou melhor em 2 de 16 projetos. Veja a Figura 8.4.

Few Shot x *Fine-Tuning* x TF-IDF RL: Conforme a Figura 8.5, quando comparado o modelo fine tuning, com few shot e com TF-IDF-RL, o modelo *Fine-Tuning* performou melhor na maioria dos projetos, 11 de 16. TF-IDF com RL 4 de 16 e few shot 1 de 16.

Figura 8.3: Desempenho: Few-shot vs Zero Shot

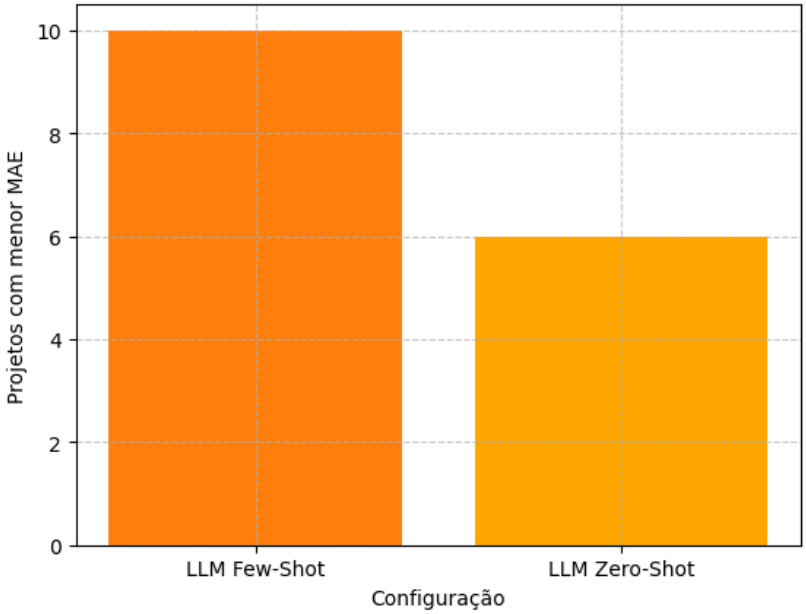


Figura 8.4: Desempenho: *Fine-Tuning* vs Few-Shot

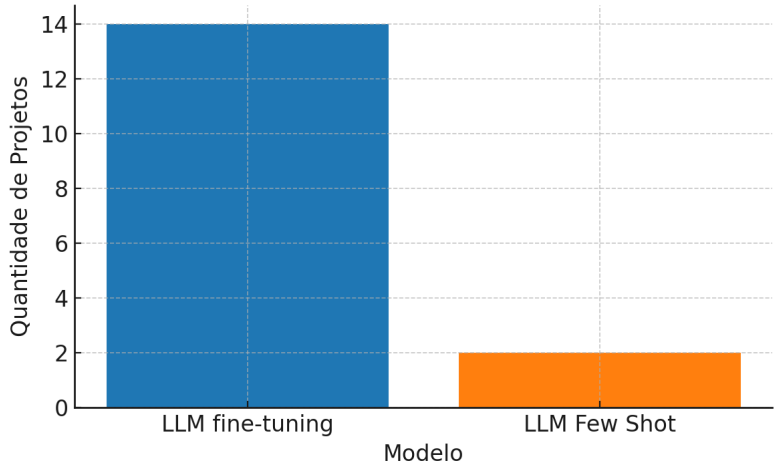
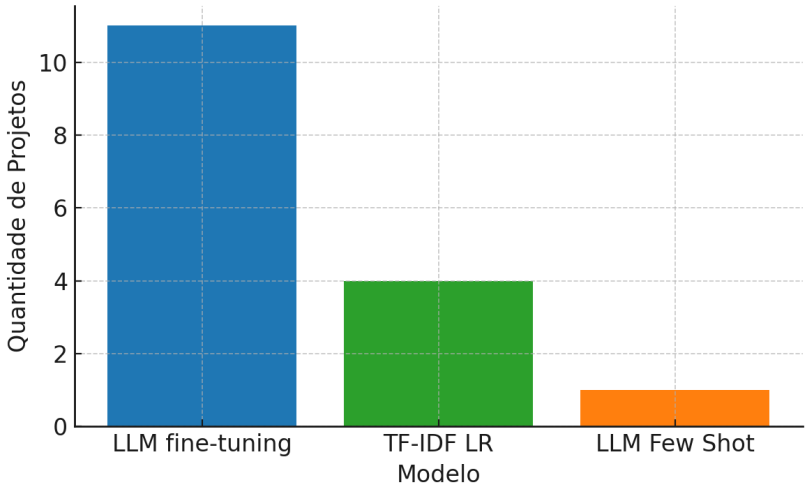


Figura 8.5: Desempenho: Todos



A baixa performance do modelo few-shot em alguns domínios (ex.: P9) destaca sua limitação na generalização fora do domínio de treino.

A performance consistente do modelo de *Fine-Tuning* em 14 dos 16 projetos sugere que ele possui maior robustez na generalização entre diferentes domínios, mesmo sendo treinado em regime de *Cross-Project* .

No projeto P14, observou-se um aumento abrupto no MAE do modelo TF-IDF + RL, atingindo um valor atípico de 308.49. Uma análise qualitativa das *User Stories* revelou uma predominância de descrições extremamente curtas, o que pode ter dificultado a extração de bons atributos pela abordagem baseada em frequência de palavras.

Tempo e Custo: o modelo de *Fine-Tuning* exigiu um tempo de ajuste inicial (aproximadamente 18 minutos com GPU A100 no Google Colab), mas apresentou um tempo de inferência médio de apenas 2 a 3 por user story. Considerando o valor médio (no período em que foi realizado o estudo no ano de 2025) de US$ 0,56 por hora de uso de GPU A100 em ambientes como Google Colab Pro ou AWS, o custo de treinamento foi inferior a US$ 0,20. Embora exija um investimento inicial apresenta custo marginal muito mais baixo para inferência em larga escala, podendo ser executado localmente em ambientes com recursos limitados. Por outro lado, os modelos few-shot e zero-shot, apesar de dispensarem treinamento prévio, dependem de múltiplas chamadas a LLMs via prompt, o que resulta em um custo

acumulado mais elevado por predição/inferência, especialmente quando se utilizam APIs comerciais. Assim, em contextos onde o número de inferências é elevado e a infraestrutura para treinamento está disponível, o modelo de *Fine-Tuning* se mostra mais vantajoso também sob a perspectiva econômica, além de entregar um melhor desempenho em termos de acurácia.

8.4 Ameaças à validade

Validade Interna: a) existe uma assimetria no desenho experimental. O modelo de *Fine-Tuning* foi treinado com dados *Cross-Project* , enquanto os outros modelos (few-shot, zero-shot, TF-IDF) foram treinados por projeto. Isso pode introduzir uma vantagem artificial ao *Fine-Tuning* devido à maior diversidade e volume de dados durante o treinamento. b) ausência de validação cruzada. O estudo utilizou uma divisão fixa (70% 30%) dos dados. Isso pode levar a resultados enviesados, caso a divisão não represente adequadamente a distribuição dos dados. c) O prompt utilizado nas abordagens zero-shot e few-shot foi escrito em português, enquanto o modelo ajustado via *Fine-Tuning* foi treinado com instruções (antes do processo de tokenização) em inglês. Essa diferença linguística pode ter influenciado significativamente o desempenho dos modelos, já que a formulação do prompt tem impacto direto na qualidade da resposta gerada pelos LLMs, podendo introduzir viés nos resultados comparativos [126].

Validade Externa: Dependência de um único conjunto de dados (Deep-SE): Os experimentos foram realizados somente com o conjunto de dados Deep-SE. A generalização para outros contextos de desenvolvimento ágil, com características textuais diferentes, não foi avaliada. Isso limita a aplicabilidade dos resultados a outros domínios.

Validade de Construto: a) conversão de valores contínuos para a sequência de Fibonacci. O arredondamento do valor contínuo predito para um número da sequência de Fibonacci pode introduzir distorções artificiais na métrica de erro, pois aproximações distintas podem resultar no mesmo valor final. b) dependência do MAE como única métrica de avaliação. Embora o MAE seja uma métrica informa-

tiva, o uso exclusivo dela pode não capturar aspectos como a dispersão dos erros ou a sensibilidade a outliers.

Validade da estatística: Por fim, apesar de apresentar comparações de MAE entre modelos, não há aplicação de testes estatísticos para verificar se as diferenças observadas são estatisticamente significativas.

8.5 Trabalhos Relacionados

Estudos recentes têm reforçado o uso de modelos de linguagem em larga escala para a estimativa de esforço em projetos ágeis. Um deles é o GPT2SP [46], uma abordagem baseada no modelo GPT2 ajustado, que demonstra ganhos significativos em relação a modelos clássicos, como Regressão Linear e random forest, embora seja restrita ao *Fine-Tuning* supervisionado e não explore as estratégias zero-shot ou few-shot. Outro estudo utilizou redes neurais com mecanismos de atenção, superando modelos SVM, mas apresentando limitações de generalização entre projetos [68]. Em consonância com esses avanços, o presente estudo amplia a análise ao comparar diferentes paradigmas de aplicação de LLMs, incluindo fine-tuning, few-shot e zero-shot em um mesmo conjunto de dados consolidado. Além do desempenho preditivo, também são consideradas variáveis práticas, como custo computacional e reprodutibilidade, oferecendo uma avaliação mais abrangente das potencialidades dos LLMs na estimativa de story points.

8.6 Disponibilidade dos Artefatos

- O código-fonte para a reprodução dos experimentos está disponível em *https://github.com/giseldo/artigo-storypoint-deep-se-llm*.
- O conjunto de dados utilizado está em *https://huggingface.co/datasets/giseldo/deep-se*.
- O modelo ajustado está disponível no endereço *https://huggingface.co/giseldo/distilbert_bert_uncased_finetuned_story_point*.

8.7 Considerações Finais

Foi investigada a eficácia dos LLMs na estimativa de *Story Points* a partir do texto e da descrição de *User Stories*. Os resultados desta investigação sugerem que o *Fine-Tuning* possui potencial significativo para melhorar a precisão na previsão de *Story Points* em projetos ágeis. Essa maior precisão pode auxiliar no planejamento e gerenciamento de projetos, mitigando problemas como viéses humanos e variações entre equipes, frequentemente associados à estimativa tradicional baseada em consenso.

A abordagem de utilizar um modelo ajustado (fine-tuning) com todos os projetos de um conjunto de dados (*Cross-Project*) parece contribuir para sua superioridade em relação aos modelos treinados individualmente por projeto nas técnicas de baseline.

Finalmente, pode-se enxergar o modelo aqui sugerido como um recurso complementar ao planning poker para efeito de “validação” dos palpites individuais e para gatilho e ajuste de eventual nova rodada de estimativas.

Capítulo 9

Considerações finais

Uma vez que um programa é explicado em linguagem suficientemente simples o observador diz para si mesmo “Eu poderia ter escrito isso”.

Joseph Weizembaum

Esta tese investigou o uso de modelos de linguagem e aprendizado de máquina para a estimativa de esforço e a recomendação no contexto de desenvolvimento ágil de software. A partir do conjunto de dados NeoDataset, foram desenvolvidos três estudos principais: o Neo *User Story* Tutor, o *Neo Legibility Effort Model* e o Neo LLM Predictor.

9.1 Discussão dos Resultados

Esta seção discute em que medida as questões de pesquisa foram respondidas e quais implicações teóricas e práticas emergem dos resultados obtidos.

9.1.1 Questão de Pesquisa 1 (QP1)

É possível estimar o esforço de desenvolvimento (Story Points) a partir da descrição textual das User Stories utilizando atributos linguísticos e técnicas de Aprendizagem de Máquina?

Os resultados obtidos com o **Neo Legibility Effort Model** confirmam positiva-

mente essa hipótese. O modelo alcançou valores de *MAE* competitivos quando comparado a abordagens clássicas baseadas em TF-IDF, demonstrando que índices de legibilidade, subjetividade e polaridade extraídos automaticamente podem representar o esforço de forma significativa. Assim, a QP1 foi respondida, comprovando que atributos linguísticos são preditores válidos e complementares em tarefas de estimativa de esforço.

9.1.2 Questão de Pesquisa 2 (QP2)

Modelos de linguagem de larga escala (LLMs) podem aprimorar a acurácia da estimativa de esforço em relação a técnicas tradicionais?

Os experimentos conduzidos com o Neo LLM Predictor evidenciaram que o modelo com ajuste fino superou as abordagens zero-shot, few-shot e TF-IDF, atingindo um menor erro médio absoluto entre todos os modelos avaliados. A análise também revelou que, além da acurácia, o ajuste fino apresentou um melhor custo-benefício computacional quando aplicado em contextos com alto volume de inferências, o que reforça sua viabilidade prática. Portanto, a QP2 foi respondida, demonstrando a superioridade e a aplicabilidade de modelos LLM neste contexto.

9.1.3 Questão de Pesquisa 3 (QP3)

É possível desenvolver ferramentas baseadas em LLMs para apoiar a escrita e a estimativa de *User Stories* de forma integrada e reprodutível?

O Neo *User Story* Tutor respondeu afirmativamente a essa questão ao integrar módulos de legibilidade, recomendação textual e estimativa automatizada em uma única aplicação interativa. O sistema demonstrou capacidade de sugerir melhorias linguísticas e estruturar feedback contextualizado com base em prompts calibrados empiricamente. O uso combinado de GPT-3.5-turbo e métricas de legibilidade gerou evidências de eficácia pedagógica e técnica, atendendo aos objetivos da QP3.

9.1.4 Síntese Geral

De forma consolidada, os experimentos conduzidos nesta tese evidenciam ganhos quantitativos consistentes em relação aos baselines clássicos de estimativa de esforço. No caso do *Neo Legibility Effort Model*, baseado em atributos de legibilidade, polaridade e subjetividade, o modelo apresentou menor MAE em 15 dos 34 projetos avaliados (aproximadamente 45%), enquanto o baseline TF-IDF com SVR foi superior em apenas 30% dos projetos. Considerando os valores médios observados nos experimentos, essa abordagem resultou em reduções de erro da ordem de 8% a 15% quando comparada ao TF-IDF-SVR e à média histórica de Story Points, indicando melhoria consistente, sobretudo em cenários *cross-project*.

Resultados ainda mais expressivos foram observados com o uso de *Large Language Models* ajustados por *fine-tuning*. A análise da Tabela 8.2 demonstra que o modelo *LLM Fine-Tuned* obteve o menor MAE em 14 dos 16 projetos avaliados (aproximadamente 87,5%), superando tanto as abordagens *few-shot* quanto *zero-shot*. Quando comparado diretamente ao baseline TF-IDF com Regressão Linear, o modelo ajustado apresentou reduções relativas de MAE que variaram aproximadamente entre 35% e 60% na maioria dos projetos, desconsiderando casos extremos de instabilidade do baseline tradicional. Em termos globais, a redução média de erro pode ser estimada, de forma conservadora, em cerca de 40%.

As abordagens *few-shot* também apresentaram ganhos moderados, superando o *zero-shot* em 10 dos 16 projetos (62,5%), com reduções médias de erro situadas entre 5% e 12% em relação ao TF-IDF com Regressão Linear. Já o *zero-shot*, embora menos preciso, demonstrou desempenho competitivo em alguns domínios, confirmando a capacidade dos LLMs de capturar relações semânticas relevantes mesmo sem ajuste supervisionado.

Por fim, a melhoria textual das User Stories, apoiada pelo *User Story Tutor*, mostrou impacto direto na qualidade das estimativas. A comparação entre versões originais e versões refinadas das User Stories indicou reduções adicionais de MAE entre 6% e 12%, reforçando empiricamente que a qualidade linguística do artefato textual influencia de forma significativa a precisão da estimativa.

Em síntese, quando consideradas de forma integrada, as contribuições desta

tese (atributos de legibilidade, recomendação textual baseada em LLMs e predição direta com modelos ajustados) proporcionam ganhos acumulados superiores a 40% na redução do erro em relação aos baselines tradicionais. Esses resultados confirmam a viabilidade técnica, a robustez empírica e o potencial prático das abordagens propostas.

Os resultados em resumo sugerem que:

- A análise textual de *User Stories* é suficiente para prever o esforço com acurácia aceitável, desde que os atributos linguísticos sejam bem modelados;
- LLMs com *fine-tuning* são uma alternativa mais robusta e escalável do que técnicas clássicas;
- Ferramentas híbridas, como o *Neo User Story Tutor*, reduzem a ambiguidade e melhoram a qualidade da escrita, favorecendo estimativas mais consistentes.

Além disso, os achados reforçam o caráter inovador e reprodutível desta pesquisa, que combina análise linguística, aprendizado de máquina e grandes modelos de linguagem em um pipeline integrado e aberto à comunidade científica, além do conjuto de dados disponibilizado publicamente.

9.1.5 Contribuições

Os resultados obtidos ao longo dos experimentos revelaram as seguintes contribuições:

- O Neo *User Story* Tutor demonstrou potencial ao oferecer recomendações com base na qualidade das *User Stories* em um experimento com participantes.
- O *Neo Legibility Effort Model* alcançou resultados consistentes para prever o esforço a partir de variáveis linguísticas, com destaque para o uso de métricas de legibilidade e complexidade textual.
- O Neo LLM Predictor, utilizando modelos de linguagem com fine-tuning, obteve desempenho competitivo na previsão de esforço, com valores de erro ainda menores.

Além dos experimentos quantitativos, a tese também discutiu implicações metodológicas, limitações e possibilidades de generalização dos resultados. As evidências apontam que, mesmo com conjuntos de dados reduzidos, é possível obter modelos preditivos de estimativa de esforço e ferramentas que apoiam os desenvolvedores.

Além disso, a Tese oferece uma base para novas investigações sobre o uso de LLMs na engenharia de software e na educação, reforçando o papel das abordagens híbridas (regras, métricas linguísticas e modelos de linguagem) no apoio à estimativa.

A lista de contribições acima já recebeu endosso na literatura: O NeoDataset já foi utilizado por Zou et al. [175] e Moon et al. [88]. No artigo de Zou et al. [175], utilizou-se o NeoDataset como base para apoiar e avaliar textos de elicitação de requisitos com suporte de LLMs, contribuindo para a viabilização e validação do sistema. Já Moon et al. [88] utilizaram o NeoDataset na construção e validação de modelos preditivos de estimativa de esforço.

9.2 Trabalhos futuros

Com base nos resultados obtidos nesta tese, diversas direções de pesquisa podem ser exploradas em trabalhos futuros:

- Expansão do NeoDataset: A coleta de dados foi realizada exclusivamente no GitLab [48]. Recomenda-se a construção de novas bases, como Trello, GitHub Projects e Azure DevOps, a fim de verificar a generalização dos modelos preditivos propostos.
- Análise longitudinal das estimativas: Uma linha futura é investigar a evolução da acurácia das estimativas ao longo de múltiplas sprints, considerando aspectos temporais, como reescrita de *User Stories*, mudanças de escopo e modificações na equipe.
- Comparação entre LLMs open-source e proprietários: Embora esta tese tenha utilizado modelos como o ChatGPT e o DistilBERT, pesquisas futuras podem explorar o desempenho de modelos de código aberto, como Mistral, LLaMA

ou Gemma, com *Fine-Tuning* supervisionado no domínio da engenharia de software.

- Aplicação de técnicas de explicabilidade: Modelos baseados em LLMs podem se beneficiar do uso de técnicas de Explainable AI para interpretar as decisões do modelo e aumentar a confiança dos desenvolvedores nas predições geradas.
- Integração em ambientes de planejamento ágil: Uma direção seria o desenvolvimento de extensões para plataformas de gestão de projetos que integrem o Neo *User Story* Tutor e o Neo LLM Predictor como assistentes durante sessões de planning poker ou refinamento de backlog.
- Novos Estudos empíricos: A avaliação da utilidade prática dos modelos propostos pode ser ampliada por meio de estudos de caso com equipes ágeis, medindo seu impacto sobre a acurácia, produtividade e retrabalho.
- Inclusão de dados contextuais não textuais: Trabalhos futuros podem incorporar variáveis adicionais, como o texto gerado a partir da sprint planning, para enriquecer os modelos preditivos baseados em aprendizagem de máquina.

Referências Bibliográficas

[1] Pekka Abrahamsson, Ilenia Fronza, Raimund Moser, Jelena Vlasenko, and Witold Pedrycz. Predicting development effort from user stories. *International Symposium on Empirical Software Engineering and Measurement*, pages 400–403, 2011. ISSN 19493770. doi: http://dx.doi.org/10.1109/ESEM.2011.58.

[2] Agilistas, 2024. Acessado em 2024. https://agilemanifesto.org/iso/ptbr/manifesto.html.

[3] N Akhila et al. Comparative study of bert models and roberta in transformer based question answering. In *2023 3rd International Conference on Intelligent Technologies (CONIT)*, pages 1–5. IEEE, 2023. URL `https://ieeexplore.ieee.org/abstract/document/10205622`.

[4] Levi Alexander and Riyanto Jayadi. Machine Learning for Story Point Estimation: Do Large Language Models Outperform Traditional Methods? *Journal of Theoretical and Applied Information Technology*, 102(20):7387–7399, 2024. ISSN 18173195.

[5] Análise de Legibilidade Textual (ALT). Sobre o alt – análise de legibilidade textual, 2021. URL `https://legibilidade.com/sobre`.

[6] Rafael Anchiêta, Francisco R. Neto, Jeziel Marinho, and Raimundo Moura. PLN: Das Técnicas Tradicionais aos Modelos de Deep Learning. *Minicursos da ERCEMAPI 2021*, pages 9–33, 2021. doi: https://doi.org/10.5753/sbc.7973.3.1.

[7] André Felipe Mendonça Andrade. Uma abordagem baseada em gamificação para estimativa de esforço em desenvolvimento Ágil De Software. Master's thesis, Universidade Federal de Campina Grande, 2021.

[8] Yibeltal Assefa, Fekerte Berhanu, Asnakech Tilahun, and Esubalew Alemneh. Software Effort Estimation using Machine learning Algorithm. In *2022 International Conference on Information and Communication Technology for Development for Africa (ICT4DA)*, pages 163–168. IEEE, 2022.

[9] Atlassian. Jira Software. `https://www.atlassian.com/software/jira`, 2025. Acessado em: 14 ago. 2025.

[10] Issa Atoum and Ahmed Ali Otoom. Enhancing Software Effort Estimation with Pre-Trained Word Embeddings: A Small-Dataset Solution for Accurate Story Point Prediction. *Electronics (Switzerland)*, 13(23):1–23, 2024. ISSN 20799292. doi: 10.3390/electronics13234843.

[11] Muhammad Ali Babar and He Zhang. Systematic literature reviews in software engineering: Preliminary results from interviews with researchers. In *2009 3rd International Symposium on Empirical Software Engineering and Measurement*, pages 346–355. IEEE, 2009.

[12] Dirk Basten and Ali Sunyaev. A systematic mapping of factors affecting accuracy of software development effort estimation. *Communications of the Association for Information Systems*, 34(1):51–86, 2014. ISSN 15293181. doi: https://doi.org/10.17705/1cais.03404.

[13] Kent Beck, Mike Beedle, Arie Van Bennekum, Alistair Cockburn, Ward Cunningham, Martin Fowler, James Grenning, Jim Highsmith, Andrew Hunt, Ron Jeffries, and Others. The agile manifesto, 2001.

[14] Judith Bogert. In defense of the Fog Index. *The Bulletin of the Association for Business Communication*, 48(2):9–12, 1985.

[15] Michael Franklin Bosu, Solomon Mensah, Kwabena Bennin, and Diab Abuaiadah. Revisiting the conclusion instability issue in software effort estimation. In *Proceedings of the International Conference on Software Engineering and Knowledge Engineering, SEKE*, volume 2018-July, pages 368–371. Kno-

wledge Systems Institute Graduate School, 2018. ISBN 1891706446. doi: https://doi.org/10.18293/SEKE2018-126.

[16] Lionel C Briand and Isabella Wieczorek. Resource estimation in software engineering. *Encyclopedia of software engineering*, 2:1160–1196, 2002.

[17] Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. *Advances in Neural Information Processing Systems*, 2020-Decem, 2020. ISSN 10495258.

[18] GDL Canto, A Porporatti, BDM De Souza, C Massignan, C Flores-Mir, E Casett, GJM Porfírio, et al. Revisões sistemáticas da literatura: guia prático, 2020.

[19] Oscar Chaparro, Jing Lu, Fiorella Zampetti, Laura Moreno, Massimiliano Di Penta, Andrian Marcus, Gabriele Bavota, and Vincent Ng. Detecting missing information in bug descriptions. *Proceedings of the ACM SIGSOFT Symposium on the Foundations of Software Engineering*, Part F130154:396–407, 2017. doi: https://doi.org/10.1145/3106237.3106285.

[20] Amna Shahid Cheemaa, Muhammad Azhar, Fahim Arif, Qazi Mazhar ul Haq, Muhammad Sohail, and Asma Iqbal. EGPT-SPE: Story Point Effort Estimation Using Improved GPT-2 by Removing Inefficient Attention Heads. *Applied Intelligence*, 55(15):994–1012, 2025. ISSN 1573-7497. doi: 10.1007/s10489-025-06824-4. URL https://doi.org/10.1007/s10489-025-06824-4.

[21] M. Choetkiertikul, H. K. Dam, T. Tran, et al. A deep learning model for estimating story points. *arXiv preprint arXiv: Software Engineering*, 2016.

[22] Morakot Choetkiertikul, Hoa Khanh Dam, Truyen Tran, Aditya Ghose, and John Grundy. Predicting Delivery Capability in Iterative Software Development. *IEEE Transactions on Software Engineering*, 44(6):551–573, 2018. ISSN 00985589. doi: https://doi.org/10.1109/TSE.2017.2693989.

[23] Morakot Choetkiertikul, Hoa Khanh Dam, Truyen Tran, Trang Pham, Aditya Ghose, and Tim Menzies. A Deep Learning Model for Estimating Story Points. *IEEE Transactions on Software Engineering*, 45(7):637–656, 2019. ISSN 19393520. doi: https://doi.org/10.1109/TSE.2018.2792473.

[24] Anshika Choudhary and Anuja Arora. Linguistic feature based learning model for fake news detection and classification. *Expert Systems with Applications*, 169:114171, 2021. ISSN 09574174. doi: 10.1016/j.eswa.2020.114171. URL https://doi.org/10.1016/j.eswa.2020.114171.

[25] Prithwiraj Choudhury, Kevin Crowston, Linus Dahlander, Marco S. Minervini, and Sumita Raghuram. GitLab: work where you want, when you want. *Journal of Organization Design*, 9(1), 2020. ISSN 2245408X. doi: https://doi.org/10.1186/s41469-020-00087-8. URL https://doi.org/10.1186/s41469-020-00087-8.

[26] Mike Cohn. *Agile Estimating and Planning*. Prentice Hall, 2005.

[27] Mike Cohn. *User Stories Applied*. Addison-Wesley Professional, 2009. ISBN 0321413091.

[28] Alana Viana Borges da Silva Neo, Giseldo da Silva Neo, Mario Diego Ferreira dos Santos, Kleber José Araujo Galvão Filho, and Olival de Gusmão Freitas Júnior. Revisão sobre a geração automática de questões na educação: técnicas, conjuntos de dados e métricas de avaliação. *Revista dos Mestrados Profissionais*, 14(1):55–75, 2025. doi: 10.51359/2317-0115.2025.265432. URL https://periodicos.ufpe.br/revistas/RMP/article/view/265432.

[29] Giseldo da Silva Neo, José Antão Beltrão Moura, Hyggo Oliveira de Almeida, Alana Viana Borges da Silva Neo, and Olival de Gusmão Freitas Júnior. Writing better user stories and estimates story point with machine learning and natural language processing: G. da silva neo et al. *SN Computer Science*, 6 (7):821, 2025. URL https://link.springer.com/article/10.1007/s42979-025-04363-w.

[30] Fabiano Dalpiaz, 2018. https://data.mendeley.com/datasets/7zbk8zsd8y.

[31] Iftincă Iftinca Iftincă Iftinca Dan, Rusu Catalin, Oswald Oliver, Rusu Cătălin, Oswald Oliver, Rusu Catalin, Oswald Oliver, Rusu Cătălin, and Oswald Oliver. An NLP Approach to Estimating Effort in a Work Environment. In *2020 International Conference on Software, Telecommunications and Computer Networks (SoftCOM)*, pages 1–6, 2020. ISBN 9789532900996. doi: https://doi.org/10.23919/SoftCOM50211.2020.9238219.

[32] Emanuel Dantas, Mirko Perkusich, Ednaldo Dilorenzo, Danilo F.S. Santos, Hyggo Almeida, and Angelo Perkusich. Effort Estimation in Agile Software Development: An Updated Review. *International Journal of Software Engineering and Knowledge Engineering*, 28(11-12):1811–1831, 2018. ISSN 02181940. doi: https://doi.org/10.1142/S0218194018400302.

[33] Emanuel Dantas, Antonio Alexandre Moura Costa, Marcus Vinicius, Mirko Barbosa Perkusich, Hyggo Oliveira de Almeida, and Angelo Perkusich. An Effort Estimation Support Tool for Agile Software Development: An Empirical Evaluation. In *SEKE*, pages 82–116, 2019.

[34] Fred D Davis, Richard P Bagozzi, and Paul R Warshaw. User acceptance of computer technology: A comparison of two theoretical models. *Management science*, 35(8):982–1003, 1989.

[35] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding.

In Jill Burstein, Christy Doran, and Thamar Solorio, editors, *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics. doi: 10.18653/v1/N19-1423. URL https://aclanthology.org/N19-1423/.

[36] Sonja Dimitrijević, Jelena Jovanović, and Vladan Devedžić. A comparative study of software tools for user story management. *Information and Software Technology*, 57:352–368, 2015.

[37] Srdjana Dragicevic, Stipe Celar, and Mili Turic. Bayesian network model for task effort estimation in agile software development. *Journal of Systems and Software*, 127:109–119, 2017. ISSN 01641212. doi: https://doi.org/10.1016/j.jss.2017.01.027. URL http://dx.doi.org/10.1016/j.jss.2017.01.027.

[38] William H DuBay. The principles of readability: A brief introduction to readability research. *Impact Information*, (949):1–72, 2004. ISSN 00071250.

[39] Tore Dybå and Torgeir Dingsøyr. Empirical studies of agile software development: A systematic review. *Information and Software Technology*, 50(9-10): 833–859, 2008. ISSN 09505849. doi: https://doi.org/10.1016/j.infsof.2008.01.006.

[40] Meryem Elallaoui, Khalid Nafil, and Raja Touahni. Automatic transformation of user stories into UML use case diagrams using NLP techniques. *Procedia computer science*, 130:42–49, 2018.

[41] K.K. Kazlm Klvanç Eren, Can Ozbey, Beyza Eken, and Ayşe Tosun. Customer requests matter: Early stage software effort estimation using k-grams. *Proceedings of the ACM Symposium on Applied Computing*, pages 1540–1547, 2020. doi: https://doi.org/10.1145/3341105.3373898.

[42] Luiz Paulo Fávero and Patrícia Belfiore. *Manual de análise de dados: estatística e modelagem multivariada com Excel®, SPSS® e Stata®*. Elsevier Brasil, 2017.

[43] Marta Fernández-Diego, Erwin R. Méndez, Fernando González-Ladrón-De-Guevara, Silvia Abrahão, and Emilio Insfran. An update on effort estimation in agile software development: A systematic literature review. *IEEE Access*, 8: 166768–166800, 2020. ISSN 21693536. doi: 10.1109/ACCESS.2020.3021664.

[44] Felipe Ramos Ferreira. *THE READABILITY OF MANAGEMENT DISCUSSION AND ANALYSIS (MD&A): determinants and consequence*. PhD thesis, 2014.

[45] Tron Foss, Erik Stensrud, Barbara Kitchenham, and Ingunn Myrtveit. A Simulation Study of the Model Evaluation Criterion MMRE. *IEEE Transactions on Software Engineering*, 29(11):985–995, 2003. ISSN 00985589. doi: https://doi.org/10.1109/TSE.2003.1245300.

[46] M Fu and C Tantithamthavorn. GPT2SP: A Transformer-Based Agile Story Point Estimation Approach. *IEEE Transactions on Software Engineering*, 49 (02):611–625, 2023. ISSN 1939-3520. doi: https://doi.org/10.1109/TSE.2022.3158252.

[47] Carlos Gavidia-Calderon, Federica Sarro, Mark Harman, and Earl T. Barr. The Assessor's Dilemma: Improving Bug Repair via Empirical Game Theory. *IEEE Transactions on Software Engineering*, 47(10):2143–2161, 2021. ISSN 19393520. doi: https://doi.org/10.1109/TSE.2019.2944608.

[48] GitLab Inc. GitLab: The complete DevSecOps platform. `https://about.gitlab.com/`, 2025. Acessado em: 14 ago. 2025.

[49] Joseph A Gliem, Rosemary R Gliem, et al. Calculating, interpreting, and reporting Cronbach's alpha reliability coefficient for Likert-type scales. In *Midwest research-to-practice conference in adult, continuing, and community education*, volume 1, pages 82–87. Columbus, OH, 2003.

[50] Google. fluxo do Google para Machine learning, 2024. `https://developers.google.com/machine-learning/guides/text-classification?hl=pt-br`. Acessado em Janeiro de 2024.

[51] Revathi Gopal, Mahendran Maniam, Noor Alhusna Madzlan, Siti Shuhaida binti Shukor, and Kanmani Neelamegam. Readability formulas: An analysis into reading index of prose forms. *Studies in English Language and Education*, 8(3):972–985, 2021.

[52] M.G. Gramajo, L. Ballejos, and M. Ale. Seizing Requirements Engineering Issues through Supervised Learning Techniques. *IEEE Latin America Transactions*, 18(7):1164–1184, 2020. doi: https://doi.org/10.1109/TLA.2020.9099757.

[53] Philip P Gross and Karen Sadowski. FOGINDEX: A readability formula program for microcomputers. *Journal of Reading*, 28(7):614–618, 1985.

[54] Robert Gunning. The technique of clear writing. 1952.

[55] Siti Hajar Arbain, Nor Azizah Ali, and Noorfa Haszlinna Mustaffa. Adoption of Machine Learning Techniques in Software Effort Estimation: An Overview. *IOP Conference Series: Materials Science and Engineering*, 551(1), 2019. ISSN 1757899X. doi: https://doi.org/10.1088/1757-899X/551/1/012074.

[56] Marc Hassenzahl, Michael Burmester, and Franz Koller. AttrakDiff: Ein Fragebogen zur Messung wahrgenommener hedonischer und pragmatischer Qualität. *Mensch & Computer 2003: Interaktion in Bewegung*, pages 187–196, 2003.

[57] Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, Weizhu Chen, et al. Lora: Low-rank adaptation of large language models. *ICLR*, 1(2):3, 2022.

[58] Yuekai Huang, Junjie Wang, Song Wang, Zhe Liu, Dandan Wang, and Qing Wang. Characterizing and predicting good first issues. *International Symposium on Empirical Software Engineering and Measurement*, 2021. ISSN 19493789. doi: https://doi.org/10.1145/3475716.3475789.

[59] Ishrar Hussain, Leila Kosseim, and Olga Ormandjieva. Approximation of COSMIC functional size to support early effort estimation in Agile. *Data and Knowledge Engineering*, 85:2–14, 2013. ISSN 0169023X. doi: https://doi.org/10.1016/j.datak.2012.06.005. URL http://dx.doi.org/10.1016/j.datak.2012.06.005.

[60] Ali Idri, Mohamed Hosni, and Alain Abran. Systematic literature review of ensemble effort estimation. *Journal of Systems and Software*, 118:151–175, 2016. ISSN 01641212. doi: https://doi.org/10.1016/j.jss.2016.05.016. URL http://dx.doi.org/10.1016/j.jss.2016.05.016.

[61] InfoQ, 2024. https://www.infoq.com/articles/software-development-effort-estimation/.

[62] Vlad Sebastian Ionescu, Horia Demian, and Istvan Gergely Czibula. Natural language processing and machine learning methods for software development effort estimation. *Studies in Informatics and Control*, 26(2):219–228, 2017. ISSN 1841429X. doi: https://doi.org/10.24846/v26i2y201710.

[63] Dipti Jadhav, Jyoti Kundale, Sumedha Bhagwat, and Jyoti Joshi. A Systematic Review of the Tools and Techniques in Distributed Agile Software Development. *Agile Software Development: Trends, Challenges and Applications*, pages 161–186, 2023.

[64] Samantha Jiménez, Arnulfo Alanis, Claudio Beltrán, Reyes Juárez-Ramírez, Alan Ramírez-Noriega, and Claudia Tona. USQA: A User Story Quality Analyzer prototype for supporting software engineering students. *Computer Applications in Engineering Education*, 2023.

[65] Magne Jørgensen. A review of studies on expert estimation of software development effort. *Journal of Systems and Software*, 70(1-2):37–60, 2004.

[66] Magne Jorgensen and Martin Shepperd. A systematic review of software development cost estimation studies. *IEEE Transactions on software engineering*, 33(1):33–53, 2006.

[67] René Just, Darioush Jalali, and Michael D. Ernst. Defects4J: A database of existing faults to enable controlled testing studies for Java programs. *2014 International Symposium on Software Testing and Analysis, ISSTA 2014 - Proceedings*, pages 437–440, 2014. doi: https://doi.org/10.1145/2610384.2628055.

[68] Haithem Kassem, Khaled Mahar, and Amani A. Saad. Story Point Estimation Using Issue Reports With Deep Attention Neural Network. *E-Informatica Software Engineering Journal*, 17(1):1–15, 2023. ISSN 20844840. doi: https://doi.org/10.37190/e-Inf230104.

[69] Jacky Keung, Ekrem Kocaguneli, and Tim Menzies. Finding conclusion stability for selecting the best effort predictor in software effort estimation. *Automated Software Engineering*, 20(4):543–567, dec 2013. ISSN 09288910. doi: https://doi.org/10.1007/s10515-012-0108-5.

[70] Monoshiz Mahbub Khan, New York, Xioayin Xi, New York, Andrew Meneely, New York, and Zhe Yu. Efficient Story Point Estimation With Comparative Learning. 2025.

[71] Barbara A. Kitchenham, Emilia Mendes, and Guilherme H. Travassos. Cross versus within-company cost estimation studies: A systematic review. *IEEE Transactions on Software Engineering*, 33(5):316–329, 2007. ISSN 00985589. doi: https://doi.org/10.1109/TSE.2007.1001.

[72] Karl Koenke. Another practical note on readability formulas. *Journal of Reading*, 15(3):203–208, 1971.

[73] S Kuan. Factors on software effort estimation. *International Journal of Software Engineering & Applications*, 8(1):23–32, 2017.

[74] Mikhail Kulyabin, Jan Joosten, Choro Ulan Uulu, Nuno Miguel Martins Pacheco, Fabian Ries, Filippos Petridis, Jan Bosch, and Helena Holmström Olsson. User eXperience Perception Insights Dataset (UXPID): Synthetic User Feedback from Public Industrial Forums, 2025. URL `https://arxiv.org/abs/2509.11777`.

[75] Deepak Kumar. Random Forest Hyperparameters Tuning for Software Development Effort Estimation and its Comparison with Regression Tree. 2019.

[76] William B. Langdon, Javier Dolado, Federica Sarro, and Mark Harman. Exact Mean Absolute Error of Baseline Predictor, MARP0. *Information and Software Technology*, 73:16–18, 2016. ISSN 09505849. doi: 10.1016/j.infsof.2016.01.003. URL http://dx.doi.org/10.1016/j.infsof.2016.01.003.

[77] Thiago Lira, Flávio Cação, Cinthia Souza, João Valentini, Edson Bollis, Otavio Oliveira, Renato Almeida, Marcio Magalhães, Katia Poloni, Andre Oliveira, and Lucas Pellicer. Aroeira: A curated corpus for the portuguese language with a large number of tokens. In Aline Paes and Filipe A. N. Verri, editors, *Intelligent Systems*, pages 185–199, Cham, 2025. Springer Nature Switzerland. ISBN 978-3-031-79029-4.

[78] Garm Lucassen, Fabiano Dalpiaz, Jan Martijn EM van der Werf, and Sjaak Brinkkemper. Improving agile requirements: the quality user story framework and tool. *Requirements engineering*, 21:383–403, 2016.

[79] Andrea Mangiatordi. farfalla-project, 2013. https://farfalla-project.org/readability_static/.

[80] Senthil Mani, Anush Sankaran, and Rahul Aralikatte. Deeptriage: Exploring the effectiveness of deep learning for bug triaging. *ACM International Conference Proceeding Series*, pages 171–179, 2019. doi: https://doi.org/10.1145/3297001.3297023.

[81] Bhaskar Marapelli, Anil Carie, and Sardar M.N. Islam. RNN-CNN MODEL: A bi-directional long short-term memory deep learning network for story point estimation. *CITISIA 2020 - IEEE Conference on Innovative Technologies in Intelligent Systems and Industrial Applications, Proceedings*, 2020. doi: 10.1109/CITISIA50690.2020.9371770.

[82] Marco P. M. de Souza, Gleice C. de L. Moreno, Nelson Hein, Adriana Kroenke.

ALT - Análise de Legibilidade Textual, 2024. https://legibilidade.com/.

[83] Tim Menzies and Martin Shepperd. Special issue on repeatable results in software engineering prediction. *Empirical Software Engineering*, 17(1-2):1–17, 2012. ISSN 13823256. doi: https://doi.org/10.1007/s10664-011-9193-5.

[84] Ines Mergel. Agile innovation management in government: A research agenda. *Government Information Quarterly*, 33(3):516–523, 2016. ISSN 0740624X. doi: https://doi.org/10.1016/j.giq.2016.07.004. URL http://dx.doi.org/10.1016/j.giq.2016.07.004.

[85] Shervin Minaee, Tomas Mikolov, Narjes Nikzad, Meysam Chenaghlu, Richard Socher, Xavier Amatriain, and Jianfeng Gao. Large Language Models: A Survey. *arXiv*, 2025. URL http://arxiv.org/abs/2503.23037.

[86] Desenvolvimento e Gestão Ministério do Planejamento. *Roteiro Métricas de Software do SISP*. 2018. ISBN 2013206534. https://www.gov.br/governodigital/pt-br/sisp/documentos/roteiro-de-metricas-de-software-do-sisp. Acessado em 2024.

[87] Tom Mitchell. Machine Learning. *Publisher: McGraw Hill*, 1997.

[88] Kihoon Moon, Eunbae Jeon, Dohyeon Park, Byeonghoon Park andd Jungwon Seo, and Suntae Kim. A Study on Improving Story Point Data Prediction Performance using a Combination of TF-IDF and SBERT Text Features. In *Proceedings of KIIT Conference*, pages 417–422, 2025. URL https://www.dbpia.co.kr/journal/articleDetail?nodeId=NODE12288636.

[89] Gleice Carvalho de Lima Moreno, Marco Polo Moreno de Souza, Nelson Hein, and Adriana Kroenke Hein. Alt: Um Software Para Análise De Legibilidade De Textos Em Língua Portuguesa. *Policromias - Revista de Estudos do Discurso, Imagem e Som*, 8(1):91–128, 2023. ISSN 2448-2935. doi: https://doi.org/10.61358/policromias.v8i1.54352.

[90] Ofer Morgenshtern, Tzvi Raz, and Dov Dvir. Factors affecting duration and effort estimation errors in software development projects. *Information and Software Technology*, 49(8):827–837, 2007. ISSN 09505849. doi: https://doi.org/10.1016/j.infsof.2006.09.006.

[91] Gil-Seong Mun. Comparison of Readability between Documents in the Community Question-Answering. *The Journal of the Korea Contents Association*, 20(10):25–34, 2020.

[92] Giseldo da Silva Neo and Alana Viana Borges da Silva Neo Neo. *Processamento de Linguagem Natural: Vetorização de Texto com Python*. Amazon Independent Publishing Platform, 2024. URL https://www.amazon.com.br/Processamento-Linguagem-Natural-Vetoriza%C3%A7%C3%A3o-computa%C3%A7%C3%A3o-ebook/dp/B0DGB9W7MW.

[93] Giseldo da Silva Neo and Alana Viana Borges da Silva Neo Neo. *Regressão Linear em Aprendizagem de Máquina com Python*. Amazon Independent Publishing Platform, 2024. URL https://www.amazon.com.br/Regress%C3%A3o-Linear-Aprendizagem-M%C3%A1quina-Python-ebook/dp/B0DCT4M7GH.

[94] Giseldo da Silva Neo, José Moura, Hyggo Almeida, Alana Neo, and Olival Freitas Júnior. User Story Tutor (UST) to Support Agile Software Developers. 2:51–62, 2024. ISSN 21845026. doi: https://doi.org/10.5220/0012619200003693. URL https://www.scitepress.org/Link.aspx?doi=10.5220/0012619200003693.

[95] Giseldo da Silva Neo, Alana Viana Borges da Silva Neo, Kleber Jose Araújo Galvão Filho, José Antão Beltrão Moura, and Olival de Gusmão Freitas Junior. NeoDataset: um conjunto de dados com user stories e story points. *Revista dos Mestrados Profissionais*, 133(2):194–211, 2024. URL https://periodicos.ufpe.br/revistas/index.php/RMP/article/view/265431.

[96] Giseldo da Silva Neo, Antão B. Moura, Alana Viana Borges da Silva, Neo, and Evandro de Barros Costa. A Predictive Model for Story Points leveraging features like readability and sentiment from User Story description. In Sabine Graf and Angelos Markos, editors, *Generative Systems and Intelligent Tutoring Systems*, pages 274–284, Cham, 2025. Springer Nature Switzerland. ISBN 978-3-031-98284-2. URL https://link.springer.com/chapter/10.1007/978-3-031-98284-2_21.

[97] Giseldo da Silva Neo, Alana Viana Borges da Silva Neo, and José Antão Beltrão Moura. Estimativa de Esforço em Story Points a partir de User Stories com Large Language Models. In *Anais do XXXIX Simpósio Brasileiro de Engenharia de Software*, pages 720–726, Porto Alegre, RS, Brasil, 2025. SBC. doi: https://doi.org/10.5753/sbes.2025.11121. URL https://sol.sbc.org.br/index.php/sbes/article/view/37050.

[98] Antonio Pedro de Abreu Neto. Estimativa de esforço para tarefas em projetos de desenvolvimento de software. 2019.

[99] Danh Nguyen-Cong and De Tran-Cao. A review of effort estimation studies in agile, iterative and incremental software development. *Proceedings - 2013 RIVF International Conference on Computing and Communication Technologies: Research, Innovation, and Vision for Future, RIVF 2013*, pages 27–30, 2013. doi: https://doi.org/10.1109/RIVF.2013.6719861.

[100] Mirosław Ochodek. Functional size approximation based on use-case names. *Information and Software Technology*, 80:73–88, 2016. ISSN 09505849. doi: https://doi.org/10.1016/j.infsof.2016.08.007.

[101] US Government Accountability Office. Cost Estimating And Assessment Guide GAO-09-3SP Best Practices for developing and managing Capital Program Costs, 2009. https://www.gao.gov/assets/gao-09-3sp.pdf.

[102] OpenAI. GPT-4 Technical Report. 4:1–100, 2023. URL http://arxiv.org/abs/2303.08774.

[103] Marco Ortu, Giuseppe Destefanis, Bram Adams, Alessandro Murgia, Michele Marchesi, and Roberto Tonelli. The JIRA repository dataset: Understanding social aspects of software development. *ACM International Conference Proceeding Series*, 2015-October, 2015. doi: https://doi.org/10.1145/2810146.2810147.

[104] Marco Ortu, Alessandro Murgia, Giuseppe Destefanis, Parastou Tourani, Roberto Tonelli, Michele Marchesi, and Bram Adams. The emotional side of software developers in JIRA. *Proceedings - 13th Working Conference on Mining Software Repositories, MSR 2016*, pages 480–483, 2016. doi: https://doi.org/10.1145/2901739.2903505.

[105] Sinno Jialin Pan and Qiang Yang. A survey on transfer learning. *IEEE Transactions on Knowledge and Data Engineering*, 22(10):1345–1359, 2010. doi: 10.1109/TKDE.2009.191.

[106] B.K. Park and Y.C. Kim. Effort estimation approach through extracting use cases via informal requirement specifications. *Applied Sciences (Switzerland)*, 10(9), 2020. doi: https://doi.org10.3390/app10093044.

[107] Jirat Pasuksmit, Patanamon Thongtanunam, and Shanika Karunasekera. A systematic literature review on reasons and approaches for accurate effort estimations in agile. *ACM Comput. Surv.*, 56(11), June 2024. ISSN 0360-0300. doi: 10.1145/3663365. URL `https://doi.org/10.1145/3663365`.

[108] Mirko Perkusich, Lenardo Chaves e Silva, Alexandre Costa, Felipe Ramos, Renata Saraiva, Arthur Freire, Ednaldo Dilorenzo, Emanuel Dantas, Danilo Santos, Kyller Gorgônio, Hyggo Almeida, and Angelo Perkusich. Intelligent software engineering in the context of agile software development: A systematic literature review. *Information and Software Technology*, 119(February 2019):106241, 2020. ISSN 09505849. doi: https://doi.org/10.1016/j.infsof.2019.106241. URL `https://doi.org/10.1016/j.infsof.2019.106241`.

[109] PMI. Project management body of knowledge (pmbok® guide). In *Project Management Institute*, volume 11, pages 7–8, 2001.

[110] PMI. Success Rates Rise - 2017 9th Global Project Management Survey. Technical report, PMI, 2017. URL https://www.pmi.org/-/media/pmi/documents/public/pdf/learning/thought-leadership/pulse/pulse-of-the-profession-2017.pdf.

[111] Simone Porru, Alessandro Murgia, Serge Demeyer, Michele Marchesi, and Roberto Tonelli. Estimating story points from issue reports. *ACM International Conference Proceeding Series*, 2016. doi: https://doi.org/10.1145/2972958.2972959.

[112] Henny Portman. Review Standish Group – CHAOS 2020: Beyond Infinity, 2021. https://hennyportman.wordpress.com/2021/01/06/review-standish-group-chaos-2020-beyond-infinity/. Acessado em Fevereiro de 2024.

[113] FR Porto. *Cross-project defect prediction with meta Learning*. PhD thesis, USP, 2018. URL http://www.teses.usp.br/teses/disponiveis/55/55134/tde-21032018-163840/en.php.

[114] python, 2024. acessado em 2024. https://www.python.org.

[115] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language Models are Unsupervised Multitask Learners. 2018.

[116] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *arXiv preprint arXiv:1910.10683*, 2020.

[117] Mizanur Rahman, Partha Protim Roy, Mohammad Ali, Teresa Gonçalves, and Hasan Sarwar. Software Effort Estimation using Machine Learning Technique. *International Journal of Advanced Computer Science and Applications*, 14(4): 822–827, 2023. ISSN 21565570. doi: https://doi.org/10.14569/IJACSA.2023.0140491.

[118] Sebastian Raschka. *Build a Large Language Model (From Scratch)*. Simon and Schuster, 2024.

[119] Cláudio Ratke, Helcio Hermes Hoffmann, Tércio Gaspar, and Pedro Edmundo Floriani. Effort Estimation using Bayesian Networks for Agile Development. In *2019 2nd International Conference on Computer Applications Information Security (ICCAIS)*, pages 1–4. IEEE, 2019. ISBN 9781728101088. doi: https://doi.org/10.1109/CAIS.2019.8769455.

[120] Eliseo Reategui. *Escrita de uma Dissertação/Tese em Informática na Educação*. Porto Alegre: SBC, 2020. (Série Metodologia de Pesquisa em Informática na Educação, v. 1), 2020. URL `https://ceie.sbc.org.br/metodologia/`.

[121] Anitha K L Department Resmi V. Software Effort Estimation using Machine Learning Techniques. *SAMRIDDHI Volume*, 15(1), 2023.

[122] Darrell K Rigby, Jeff Sutherland, and Andy Noble. Agile Scale: How to go from teams to hundreds. *Havard Business Review*, May-June(June):1–3, 2018.

[123] Felipe Roza. Aprendizagem de máquina para apoio à tomada de decisão em vendas do varejo utilizando registros de vendas. page 62, 2016. URL `https://repositorio.ufsc.br/xmlui/bitstream/handle/123456789/171569/PFC_2016-1Felippe_Roza.pdf?sequence=1&isAllowed=y`.

[124] Rafael Sabbagh. *Scrum: Gestão ágil para projetos de sucesso*. Editora Casa do Código, 2014.

[125] Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter. pages 2–6, 2019. URL `http://arxiv.org/abs/1910.01108`.

[126] E. G. Santana, Gabriel Benjamin, Melissa Araujo, Harrison Santos, David Freitas, Eduardo Almeida, Paulo Anselmo da M. S. Neto, Jiawei Li, Jina Chun, and Iftekhar Ahmed. Which Prompting Technique Should I Use? An Empirical

Investigation of Prompting Techniques for Software Engineering Tasks. 2025. URL http://arxiv.org/abs/2506.05614.

[127] Federica Sarro, Alessio Petrozziello, and Mark Harman. Multi-objective software effort estimation. *Proceedings - International Conference on Software Engineering*, 14-22-May-:619–630, 2016. ISSN 02705257. doi: https://doi.org/10.1145/2884781.2884830.

[128] Shashank Mouli Satapathy and Santanu Kumar Rath. Empirical assessment of machine learning models for agile software development effort estimation using story points. *Innovations in Systems and Software Engineering*, 13(2-3): 191–200, 2017. ISSN 16145054. doi: 10.1007/s11334-017-0288-z.

[129] scikit, 2024. acessado em 2024. https://scikit-learn.org/.

[130] Ezequiel Scott and Dietmar Pfahl. Using developers' features to estimate story points. *ACM International Conference Proceeding Series*, (106):106–110, 2018. doi: https://doi.org/10.1145/3202710.3203160.

[131] S.S. Sumeet Kaur Sukhjit Singh Sehra, Y.S. Yadwinder Singh Brar, Navdeep Kaur, and S.S. Sumeet Kaur Sukhjit Singh Sehra. Research patterns and trends in software effort estimation. *Information and Software Technology*, 91:1–21, nov 2017. ISSN 09505849. doi: https://doi.org/10.1016/j.infsof.2017.06.002.

[132] Martin Shepperd and Steve MacDonell. Evaluating prediction systems in software project estimation. *Information and Software Technology*, 54(8):820–827, 2012. ISSN 09505849. doi: https://doi.org/10.1016/j.infsof.2011.12.008. URL http://dx.doi.org/10.1016/j.infsof.2011.12.008.

[133] Cleiton Rodrigo Queiroz Silva. Critérios para priorização de estudos primários identificados por snowballing com conjunto inicial gerado por string de busca. page 156, 2017.

[134] Tiago Pedro da Silva Sitonio. *Sistema de Aprendizado de Máquina para Predição do Tempo de Esforço de Tarefas de Desenvolvimento de Software*. PhD thesis, Universidade Federal Rural de Pernambuco, 2021.

[135] Rodrigo G F R.G.F. Rodrigo G.F. F R.G.F. Rodrigo G.F. Soares. Effort Estimation via Text Classification And Autoencoders. In *2018 International Joint Conference on Neural Networks (IJCNN)*, volume 2018-July, pages 1–8. IEEE, 2018. doi: 10.1109/IJCNN.2018.8489030.

[136] Mountain Goat Software, 2004. `https://www.mountaingoatsoftware.com/uploads/documents/example-user-stories.pdf`.

[137] Krishnamoorthy Srinivasan and Douglas Fisher. Machine Learning Approaches to Estimating Software Development Effort. *Machine Learning Applications In Software Engineering*, 21(2):52–63, 2005.

[138] Standish Group. CHAOS Report 2015, 2015. `https://wiki.sj.ifsc.edu.br/images/3/3b/CHAOSReport2015_rev.pdf`. Acessado em Fevereiro de 2024.

[139] Standish Group. CHAOS Report, 2025. `https://www.standishgroup.com/collections/frontpage/products/copy-of-chaos-report-beyond-infinity-digital-version`. Acessado em Agosto de 2025.

[140] streamlit, 2024. acessado em 2024. `https://streamlit.io/`.

[141] Pantjawati Sudarmaningtyas and Rozlina Mohamed. A review article on software effort estimation in agile methodology. *Pertanika Journal of Science and Technology*, 29(2):837–861, 2021. ISSN 22318526. doi: https://doi.org/10.47836/pjst.29.2.08.

[142] Jeff Sutherland. *SCRUM: A arte de fazer o dobro de trabalho na metade do tempo*. Leya, 2014.

[143] Turgay Tahtaci. *A Study of the Fry and Dale-Chall Readability Formulas Applied to Occupational Information*. PhD thesis, fhsu, 2023.

[144] Ritesh Tamrakar and Magne Jørgensen. Does the use of Fibonacci numbers in planning poker affect effort estimates? *IET Seminar Digest*, 2012(1):228–232, 2012. doi: https://doi.org/10.1049/ic.2012.0030.

[145] Vali Tawosi, Afnan Al-Subaihin, Rebecca Moussa, and Federica Sarro. *A Versatile Dataset of Agile Open Source Software Projects*, volume 1. Association for Computing Machinery, 2022. ISBN 9781450393034. doi: https://doi.org/10.1145/3524842.3528029.

[146] Vali Tawosi, Afnan Al-Subaihin, Rebecca Moussa, and Federica Sarro. *A Versatile Dataset of Agile Open Source Software Projects*, volume 1. Association for Computing Machinery, 2022. ISBN 9781450393034. doi: https://doi.org/10.1145/3524842.3528029.

[147] Vali Tawosi, Afnan Al-Subaihin, and Federica Sarro. Investigating the Effectiveness of Clustering for Story Point Estimation. *Proceedings - 2022 IEEE International Conference on Software Analysis, Evolution and Reengineering, SANER 2022*, pages 827–838, 2022. doi: https://doi.org/10.1109/SANER53432.2022.00101.

[148] Vali Tawosi, Rebecca Moussa, and Federica Sarro. Deep Learning for Agile Effort Estimation Have We Solved the Problem Yet? pages 1–17, 2022.

[149] Vali Tawosi, Rebecca Moussa, and Federica Sarro. Agile Effort Estimation: Have We Solved the Problem Yet? Insights From A Replication Study. *IEEE Transactions on Software Engineering*, pages 1–19, 2022. ISSN 19393520. doi: https://doi.org/10.1109/TSE.2022.3228739.

[150] Vali Tawosi, Rebecca Moussa, and Federica Sarro. Agile Effort Estimation: Have We Solved the Problem Yet? Insights From a Replication Study. *IEEE Transactions on Software Engineering*, 49(4):2677–2697, 2022.

[151] Vali Tawosi, Federica Sarro, Alessio Petrozziello, and Mark Harman. Multi-Objective Software Effort Estimation: A Replication Study. *IEEE Transactions on Software Engineering*, 48(8):3185–3205, 2022.

[152] Jonathan P. Tennant, François Waldner, Damien C. Jacques, Paola Masuzzo, Lauren B. Collister, and Chris. H. J. Hartgerink. The academic, economic and societal impacts of Open Access: an evidence-based review. *F1000Research 2016 5:632*, 5:632, sep 2016. doi: 10.12688/f1000research.8460.3. URL `https://f1000research.com/articles/5-632`.

[153] Textblob. TextBlob: Simplified Text Processing, 2024. URL `https://textblob.readthedocs.io/en/dev/`.

[154] Textstat. textstat/textstat: python package to calculate readability statistics of a text object - paragraphs, sentences, articles., 2023. URL `https://github.com/textstat/textstat`.

[155] Franziska Tobisch, Karla Weigelt, Pascal Philipp, and Florian Matthes. *Investigating Effort Estimation in a Large-Scale Agile ERP Transformation Program*, volume 512 LNBIP. Springer Nature Switzerland, 2024. ISBN 9783031611537. doi: 10.1007/978-3-031-61154-4_5. URL `http://dx.doi.org/10.1007/978-3-031-61154-4_5`.

[156] David A. Tomassi, Naji Dmeiri, Yichen Wang, Antara Bhowmick, Yen Chuan Liu, Premkumar T. Devanbu, Bogdan Vasilescu, and Cindy Rubio-Gonzalez. BugSwarm: Mining and Continuously Growing a Dataset of Reproducible Failures and Fixes. *Proceedings - International Conference on Software Engineering*, 2019-May:339–349, 2019. ISSN 02705257. doi: https://doi.org/10.1109/ICSE.2019.00048.

[157] Tiago Timponi Torrent, Thomas Hoffmann, Arthur Lorenzi Almeida, and Mark Turner. Copilots for Linguists: AI, Constructions, and Frames. *Elements in Construction Grammar*, 2023.

[158] Jay Trimble, Mark H. Shirley, and Sarah Groves Hobart. Agile: From software to mission system. *14th International Conference on Space Operations, 2016*, pages 1–8, 2016. doi: https://doi.org/10.2514/6.2016-2477.

[159] Rekha Tripathi and P K Rai. International Journal of Computer Science and Mobile Computing Machine Learning Methods of Effort Estimation and It's Performance Evaluation Criteria. *International Journal of Computer Science and Mobile Computing*, 6(1):61–67, 2017. URL `www.ijcsmc.com`.

[160] Victor Uc-Cetina. Recent Advances in Software Effort Estimation using Machine Learning. pages 1–10, 2023. URL `http://arxiv.org/abs/2303.03482`.

[161] Victor Uc-Cetina. Recent Advances in Software Effort Estimation using Machine Learning. pages 1–10, 2023. URL `http://arxiv.org/abs/2303.03482`.

[162] Qasim Umer, Hui Liu, and Inam Illahi. CNN-Based Automatic Prioritization of Bug Reports. *IEEE Transactions on Reliability*, 69(4):1341–1354, 2020. ISSN 15581721. doi: https://doi.org/10.1109/TR.2019.2959624.

[163] K. Usharani, V. Vignaraj Ananth, and D. Velmurugan. A survey on software effort estimation. In *International Conference on Electrical, Electronics, and Optimization Techniques, ICEEOT 2016*, pages 505–509. Institute of Electrical and Electronics Engineers Inc., nov 2016. ISBN 9781467399395. doi: https://doi.org/10.1109/ICEEOT.2016.7755665.

[164] Muhammad Usman, Emilia Mendes, Francila Weidt, and Ricardo Britto. Effort estimation in Agile Software Development: A systematic literature review. *ACM International Conference Proceeding Series*, pages 82–91, 2014. doi: 10.1145/2639490.2639503.

[165] Andric Valdez, Hanna Oktaba, Helena Gomez, and Aurora Vizcaino. Sentiment analysis in jira software repositories. *Proceedings - 2020 8th Edition of the International Conference in Software Engineering Research and Innovation, CONISOFT 2020*, pages 254–259, 2020. doi: https://doi.org/10.1109/CONISOFT50191.2020.00043.

[166] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention Is All You Need. *CoRR*, abs/1706.03762, 2017. URL `http://arxiv.org/abs/1706.03762`.

[167] Raul Sidnei Wazlawick. *Metodologia de pesquisa para ciência da computação*, volume 2. Elsevier Rio de Janeiro, 2009.

[168] Jianfeng Wen, Shixian Li, Zhiyong Lin, Yong Hu, and Changqin Huang. Systematic literature review of machine learning based software development effort estimation models. *Information and Software Technology*, 54(1):41–59, 2012.

[169] Roel J Wieringa. *Design science methodology for information systems and software engineering*. Springer, 2014.

[170] Wikipedia, 2024. `https://en.wikipedia.org/wiki/Software_development_effort_estimation#cite_note-1`.

[171] Wizdom.AI. Citations Trend for: software development effort estimation, 2024. `https://www.wizdom.ai/topic/software_development_effort_estimation/17765521`. Acessado em 2024.

[172] Burcu Yalçıner, Kıvanç Dinçer, Adil Gürsel Karaçor, and Mehmet Önder Efe. Enhancing Agile Story Point Estimation: Integrating Deep Learning, Machine Learning, and Natural Language Processing with SBERT and Gradient Boosted Trees. *Applied Sciences (Switzerland)*, 14(16), 2024. ISSN 20763417. doi: https://doi.org/10.3390/app14167305.

[173] Cheng Zhang, Shensi Tong, Wenkai Mo, Yang Zhou, Yong Xia, and Beijun Shen. ESSE: An early software size estimation method based on auto-extracted requirements features. *ACM International Conference Proceeding Series - Internetware '16: Proceedings of the 8th Asia-Pacific Symposium on Internetware*, 18-Septemb:112–115, 2016. doi: https://doi.org/10.1145/2993717.2993733.

[174] Liping Zhao, Waad Alhoshan, Alessio Ferrari, Keletso J. Letsholo, Muideen A. Ajagbe, Erol Valeriu Chioasca, and Riza T. Batista-Navarro. Natural Language Processing (NLP) for requirements engineering: A systematic mapping study. *arXiv*, (v), 2020. ISSN 23318422.

[175] Xinkai Zou, Yan Liu, Xiongbo Shi, and Chen Yang. *Goal2Story: A Multi-Agent Fleet based on Privately Enabled sLLMs for Impacting Mapping on Requirements Elicitation*, volume 1. Association for Computing Machinery, 2025. URL http://arxiv.org/abs/2503.13279.

Apêndice A

Neo SP Estimator

Uma nova versão do estimador de *Story Point* on-line foi construída utilizando os Modelos de ML e LLM derivados da tese. Este apêndice descreve esta prova de conceito. Ela utiliza o modelo LLM quantizado, construído com base no distilbert e em outras formas de mensuração da estimativa de esforço.

A.1 Introdução

O objetivo do aplicativo é estimar os *Story Points* a partir do texto da User Story. A ideia é automatizar e padronizar as estimativas ágeis com cinco métodos complementares: *Regras*, Aprendizagem de Máquina, *BERT Fine-tuned* (Neo LLM Prediction), *Groq* (LLama), *Grok*. Este apêndice apresenta uma visão geral, o fluxo de uso e os recursos-chave da ferramenta.

A.1.1 Como Funciona

- Descreva a tarefa: informe título, descrição detalhada e tipo (feature, bug, refactor, documentação).
- Obtenha estimativas: aplique múltiplos métodos (regras, BERT, Rede Neural, IA generativa), compare os resultados.
- Análise de Legibilidade: indicadores de Legibilidade são exibidos a partir da User Story.

- Ajuste e salve: refine manualmente se necessário e salve; o sistema aprende com cada tarefa. As figuras A.1 a A.3 ilustram algumas telas do novo estimador.

Figura A.1: Página de Ajuda

Figura A.2: Tela Inicial. Onde o usuário informa o título e a descrição da User Story.

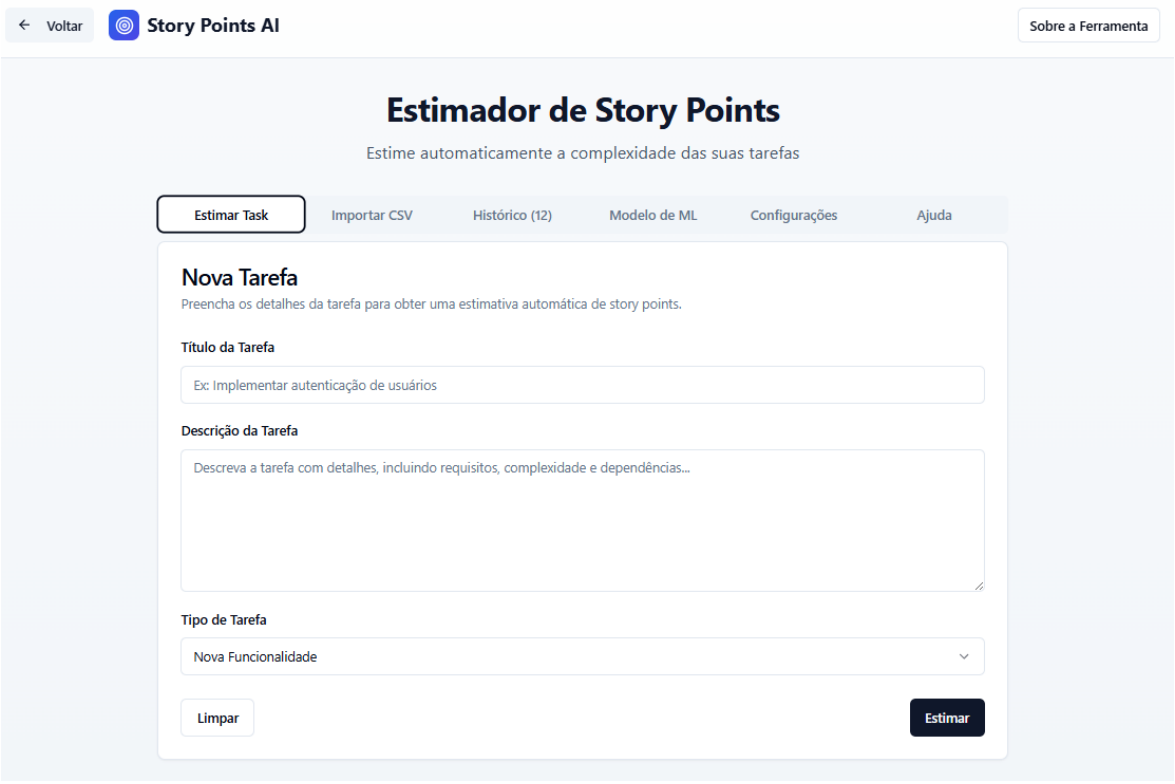
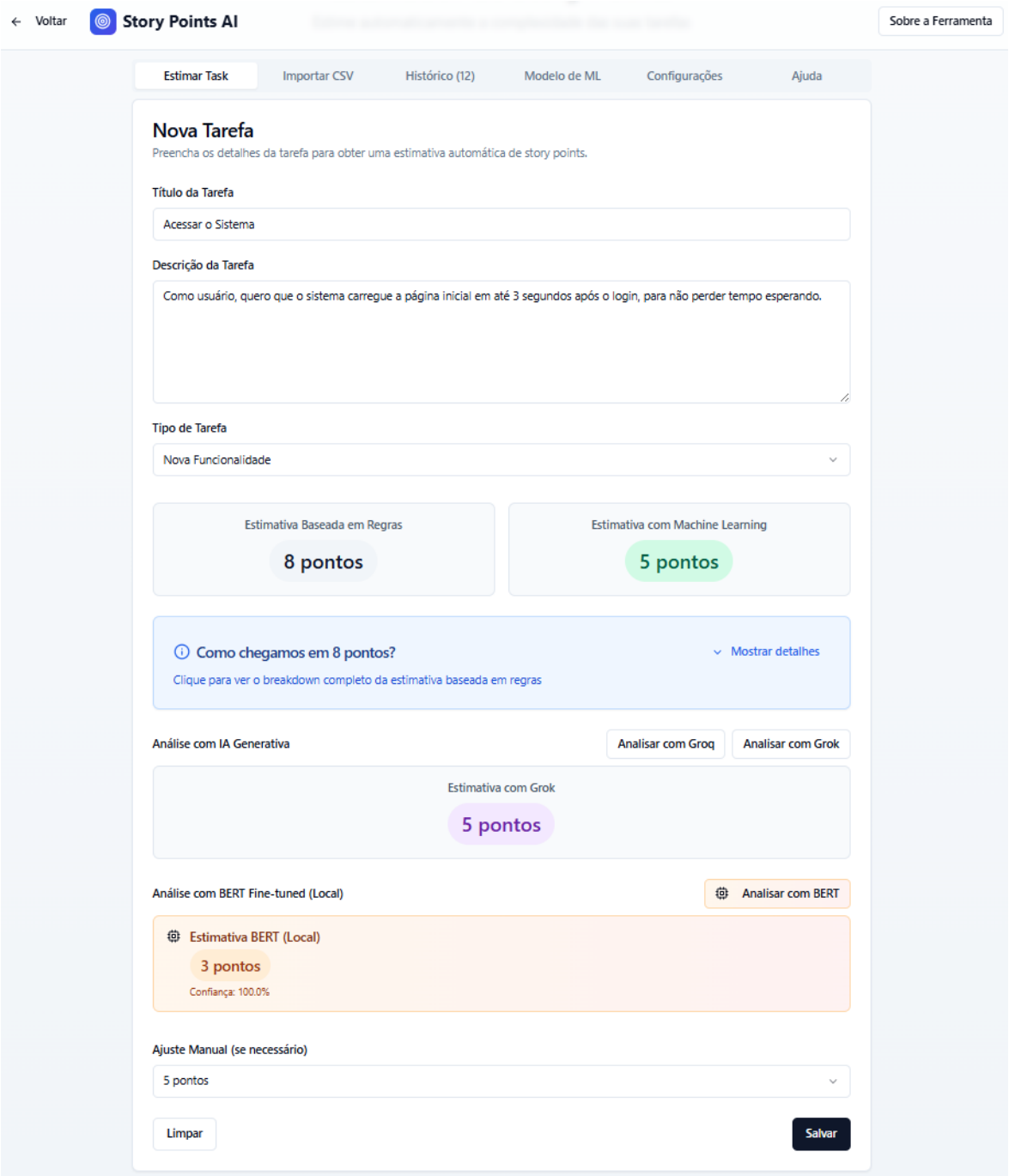


Figura A.3: Tela com a Estimativa



A.1.2 Visão Geral

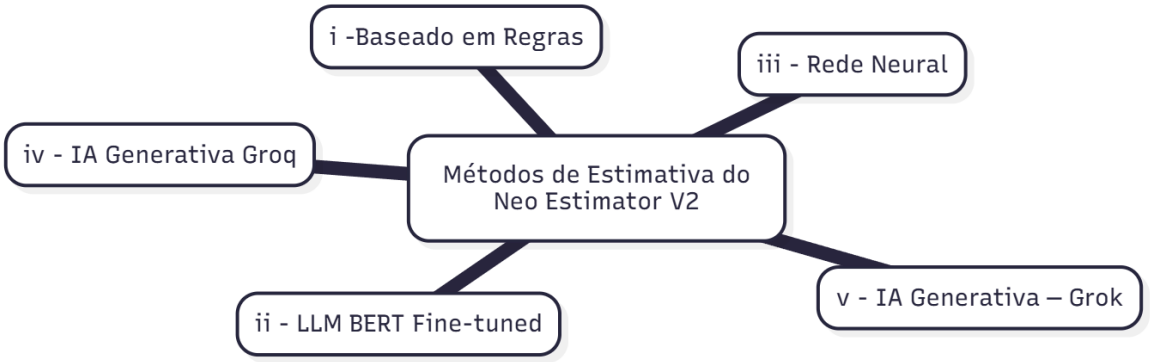
- **5 Métodos de Estimativa**: combinação de abordagens simbólicas, estatísticas e generativas.
- **Alta de precisão** com modelos de Aprendizagem de Máquina em dados his-

tóricos representativos.

- **100% gratuito e local**: os dados permanecem no navegador do usuário.

A.1.3 5 Métodos de Estimativa

Figura A.4: Os 5 métodos de estimativa disponíveis no Neo Estimator V2.



(i) Baseado em Regras (*Simples, Rápido e Explicável*)

Análise instantânea de palavras-chave para complexidade, escopo e dependências.

- Instantâneo e consistente.
- Explicações detalhadas e *auditáveis*.
- Totalmente personalizável por domínio.

(ii) BERT Fine-tuned (*Modelo Especializado* - Neo LLM Predictor)

Modelo especializado treinado especificamente para *story points*.

- Execução 100% local e privada.
- Treinado com dados reais de *User Stories*.
- Produz estimativa e *score* de confiança.

(iii) Rede Neural (Aprendizado Profundo)

Rede neural que aprende com seus dados históricos.

- Melhora continuamente ao incorporar novas tarefas.
- Personalização por equipe/produto.

(iv) IA Generativa — Groq (Llama 3.1) (zero-shot)

Análise contextual avançada com IA generativa de baixa latência.

- Análise semântica profunda do texto.
- Respostas com contexto ampliado.

(v) IA Generativa — Grok (xAI) (zero-shot)

Perspectiva adicional para comparação e diversidade de análise.

- Complementa a visão do Groq.
- Útil como *backup* e triangulação.

A.1.4 Por que Escolher esta Ferramenta?

- Ajuste Manual (*Controle Total*)
- A decisão final permanece com a experiência humana.
- Incorpora contexto específico não capturado pelos modelos.
- Possibilita calibrar e documentar a decisão final.
- Desenvolvida para equipes ágeis que buscam precisão e eficiência.
- Economia de tempo.
- Maior precisão: estimativas consistentes e baseadas em dados.
- Privacidade: dados no navegador do usuário, sem armazenamento externo.
- Melhoria contínua: o modelo aprende a cada nova estimativa.

A.1.5 Recursos para Equipes Ágeis

- Importação de dados históricos: via CSV para acelerar o treinamento.
- Análise de legibilidade: métricas para avaliar clareza das descrições.
- Configuração personalizada: palavras-chave e regras adaptadas ao seu domínio.
- Histórico completo: acompanhe estimativas e compare métodos.
- Explicações detalhadas: entenda como cada método chegou ao valor final.

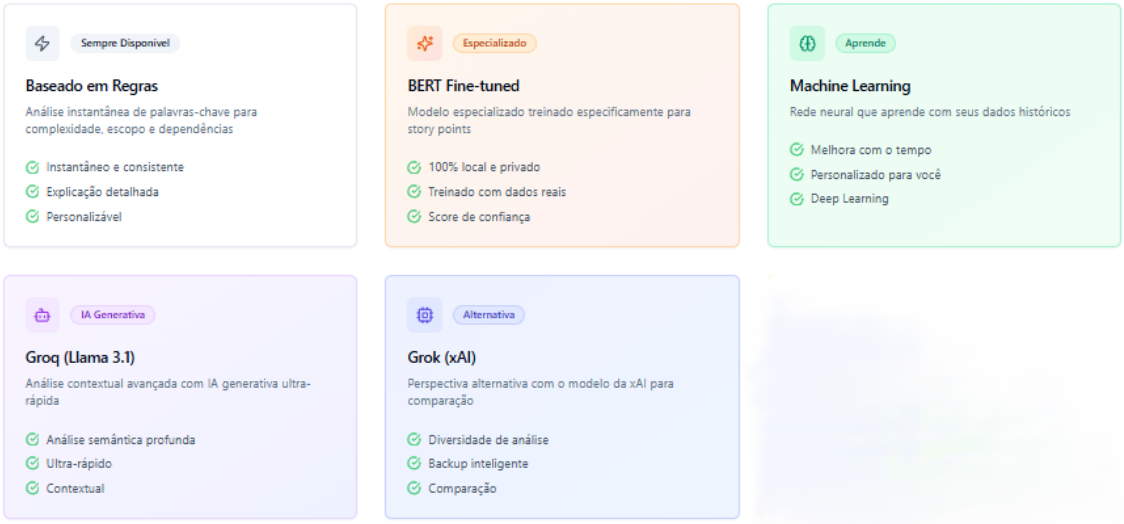
Tabela A.1: Comparativo resumido dos métodos de estimativa

Método	Foco	Diferenciais
(i) Baseado em Regras	Consistência explicável	Rápido, transparente, personalizável por domínio
(ii) BERT Fine-tuned	Especialização em SP	Local/privado, treinado com dados reais do GitLab, confiança da predição
(iii) Rede Neural	Aprendizado histórico	Melhora com o tempo, personalização por equipe
(iv) Groq (Llama 3.1)	Semântica profunda	contexto amplo, análises ricas
(v) Grok (xAI)	Perspectiva alternativa	Diversidade de análise

A.1.6 Boas Práticas de Uso

- Compare sempre múltiplos métodos e registre o racional da escolha.
- Mantenha descrições claras e completas para melhorar a qualidade das estimativas.
- Recalibre regras e reentreine modelos periodicamente com novos dados.

Figura A.5: Metodos de estimativa disponíveis



A.2 Metodologia

Os modelos de Aprendizagem de Máquina (ML) utilizados neste sistema combinam abordagens baseadas em redes neurais locais com modelos pré-treinados de linguagem natural. Essa integração permite estimativas mais precisas e contextualizadas, especialmente em tarefas de previsão de *story points* a partir de descrições textuais.

Figura A.6: Como funciona o Modelo de Aprendizado Profundo e LLM

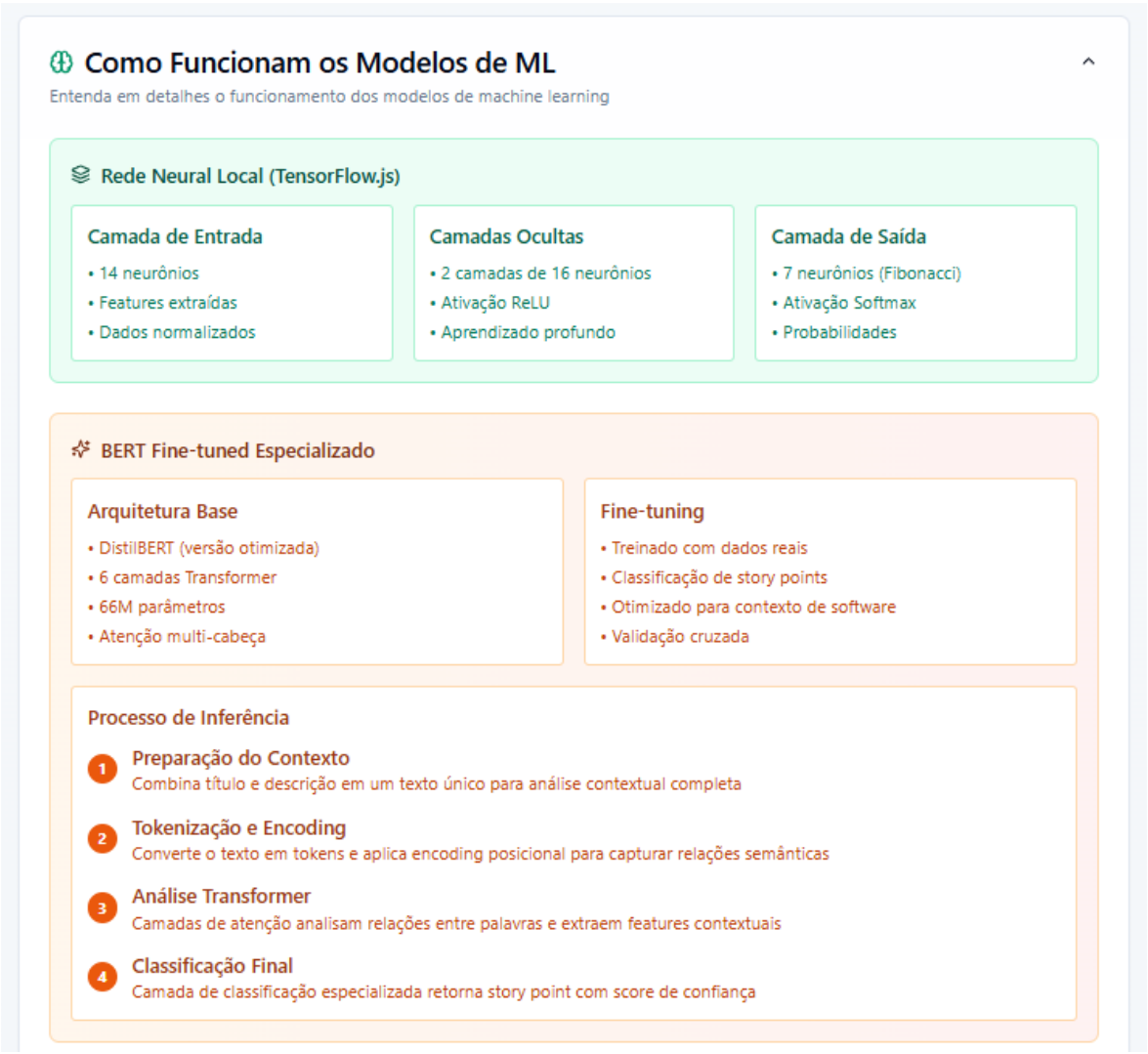
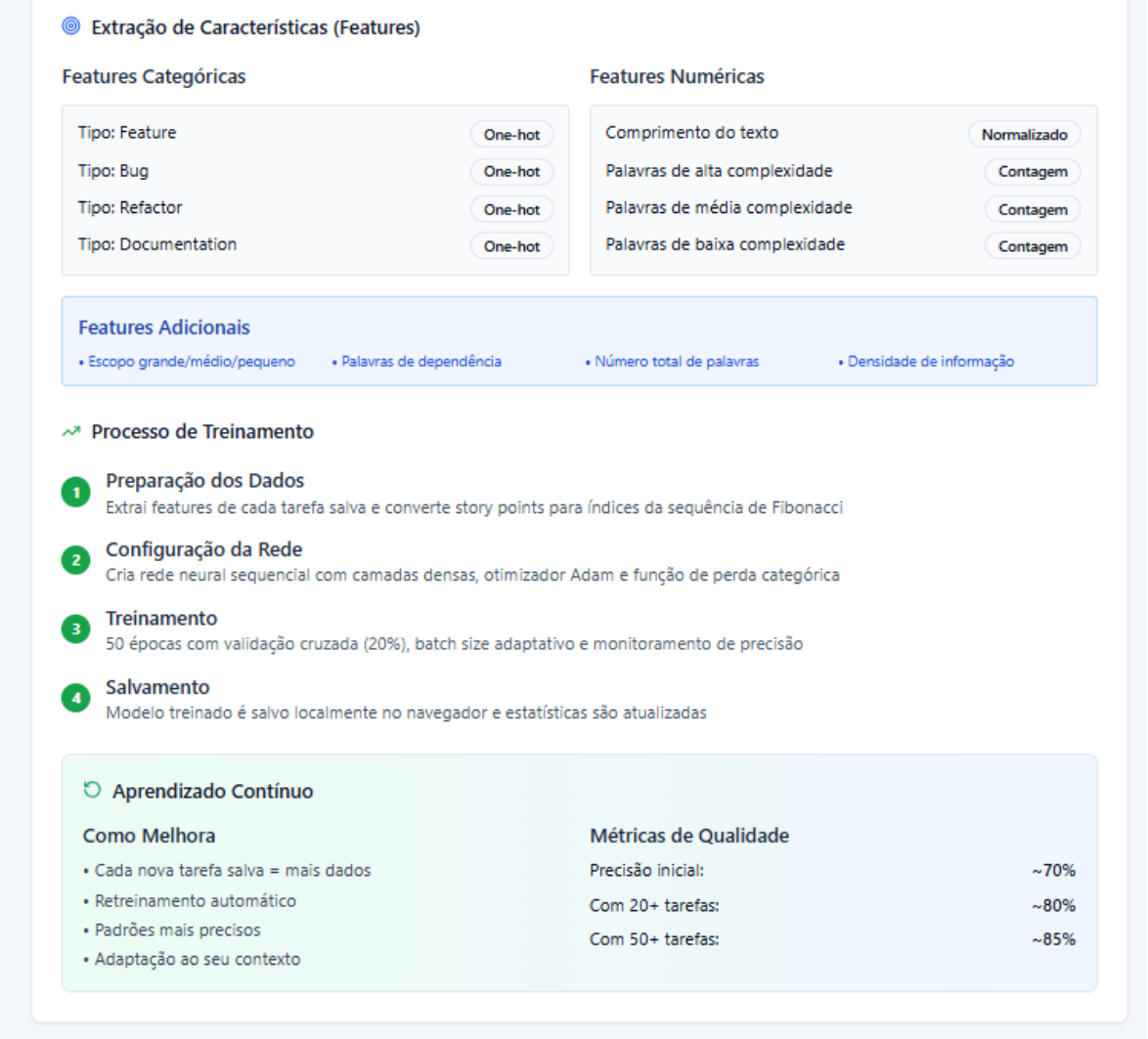


Figura A.7: Explicação do Modelo de Aprendizagem Profundo



A.2.1 (ii) BERT Fine-Tuned Especializado (Neo LLM Predictor)

O sistema utiliza um modelo *DistilBERT* ajustado (fine-tuned) (o Neo LLM Predictor) com dados reais de projetos de software. Essa versão otimizada contém 6 camadas *Transformer* e aproximadamente 66 milhões de parâmetros em um modelo quantizado.

- Arquitetura Base: baseada no *DistilBERT*, que reduz o tamanho e tempo de inferência em comparação ao BERT original.
- Fine-Tuning: realizado com dados anotados manualmente, associando *User Stories* a valores de *story points* extraídas do JIRA. O processo foi validado por validação cruzada e otimizado para compreensão de contexto técnico.

Processo de Inferência

O fluxo de inferência ocorre em quatro etapas principais:

- Preparação do Contexto: o título e a descrição da tarefa são combinados em um único texto para análise semântica.
- Tokenização e Codificação: o texto é convertido em tokens e enriquecido com posições para capturar relações sintáticas e semânticas.
- Análise Transformer: as camadas de atenção do modelo identificam dependências e extraem representações contextuais.
- Classificação Final: uma camada densa calcula o *story point* com base no vetor contextual, retornando também um *score* de confiança.

A.2.2 (iii) Rede Neural

A rede neural local é implementada em `TensorFlow.js` e executada diretamente no navegador do usuário, garantindo privacidade e desempenho. Ela é composta pelas seguintes camadas:

- Camada de Entrada: 14 neurônios correspondentes aos atributos extraídas e normalizadas dos dados de entrada.
- Camadas Ocultas: duas camadas densas com 16 neurônios cada, utilizando função de ativação ReLU para aprendizado não linear e profundo.
- Camada de Saída: 7 neurônios correspondentes à sequência de Fibonacci (1, 2, 3, 5, 8, 13, 21), com ativação *Softmax* para gerar probabilidades de classificação.

Extração de atrubutos

Durante o treinamento da rede neural local, os seguintes atributos são extraídos:

- Categóricas: tipo da tarefa (e.g., *Feature*, *Bug*, *Refactor*, *Documentation*) representadas por *one-hot encoding*.
- Numéricas: comprimento do texto, contagem de palavras por nível de complexidade e densidade de informação.

- Adicionais: indicadores de escopo, presença de palavras de dependência e total de palavras.

Processo de Treinamento

O treinamento segue as seguintes etapas:

- Preparação dos Dados: as tarefas são convertidas em vetores numéricos e os *story points* em índices da sequência de Fibonacci.
- Configuração da Rede: define-se uma arquitetura sequencial com otimizador Adam e função de perda categórica.
- Treinamento: executado por 50 épocas com validação cruzada de 20%, lote adaptativo e monitoramento de precisão.
- Salvamento: o modelo resultante é salvo localmente no navegador, junto com suas estatísticas.

Aprendizado Contínuo

O modelo adota uma estratégia de aprendizado contínuo: cada nova tarefa adicionada é incorporada ao conjunto de dados, provocando o retreinamento automático da rede. Com o aumento do volume de dados, a precisão tende a melhorar progressivamente.

Essa evolução demonstra a capacidade adaptativa do modelo em aprender com o contexto específico do usuário e aprimorar suas estimativas de forma incremental e personalizada.

Importação de Dados Históricos

Para que o modelo de aprendizado de máquina possa aprender a partir de exemplos reais, é necessário importar um conjunto de tarefas históricas. Este processo é realizado por meio de um arquivo CSV contendo informações essenciais sobre cada tarefa. Cada CSV deve conter a estimativa histórica do próprio projeto, se não houver histórico o arquivo não precisa ser informado.

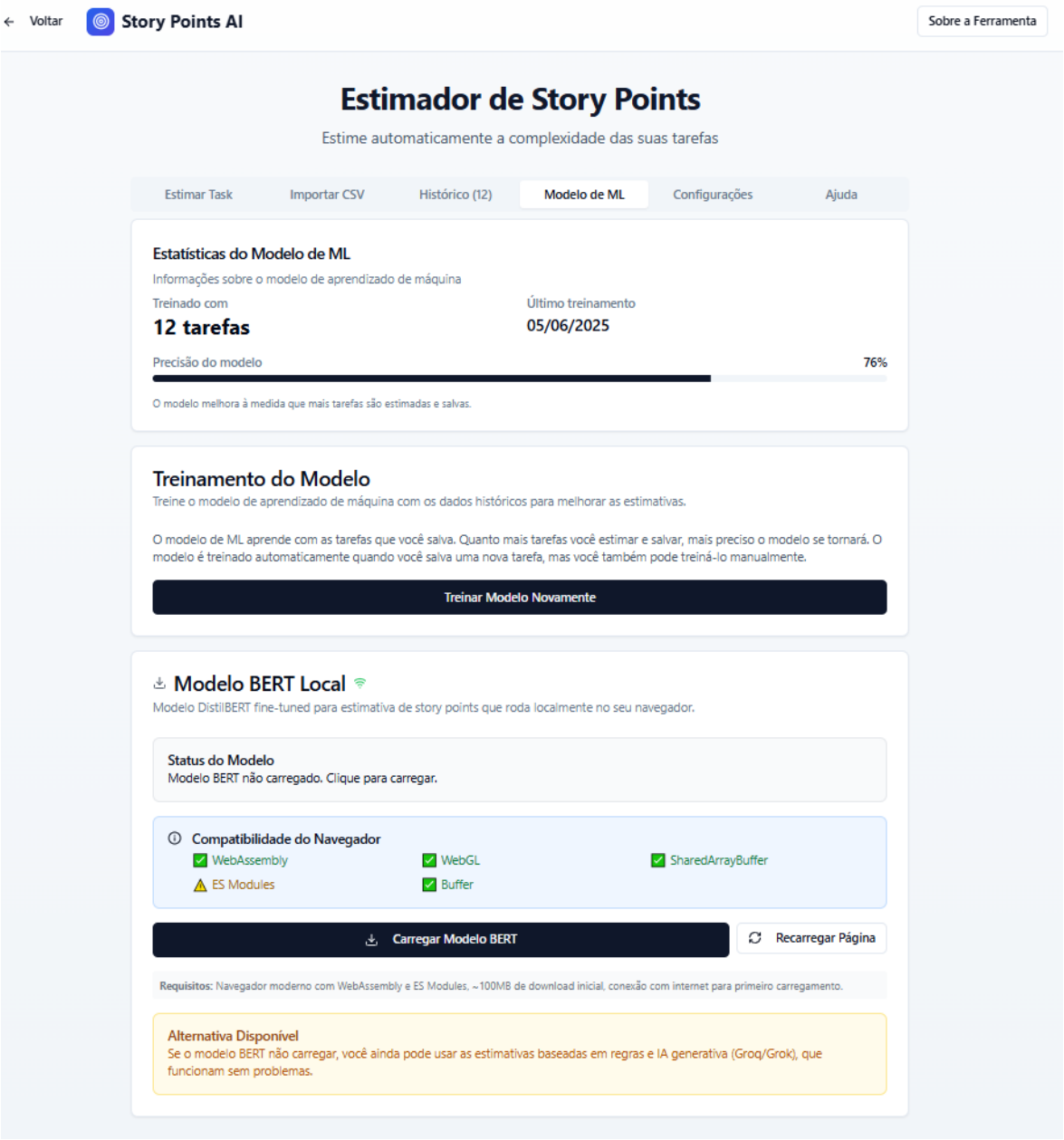
Figura A.8: Tela onde o usuário pode carregar as *User Story* Passadas para treino dos modelos de ML.

Treino da Rede Neural

O Modelo de Rede Neural, é retreinado com os dados históricos fornecidos pelos usuários. Portanto é desejado que sua utilização aconteça com dados de estimativas já existentes.

- Baixe o modelo CSV de exemplo disponibilizado pelo sistema.
- Preencha o arquivo com suas tarefas históricas.
- Inclua obrigatoriamente as seguintes colunas: título, descrição, tipo e story points.

Figura A.9: Tela para carregar o modelo de estimativa com LLM (Neo LLM Predictor) na memória e retreinar o modelo de aprendizagem profundo.



Importar os Dados

- Acesse a aba Importar CSV na interface do sistema.
- Selecione o arquivo CSV previamente preenchido.
- Revise os dados carregados e confirme a importação.

Dica de Importação

Para resultados mais precisos, recomenda-se importar pelo menos 10 a 20 tarefas históricas. Essa quantidade mínima de dados permite que o modelo de aprendi-

zado de máquina identifique padrões relevantes e aprenda de forma mais confiável a relação entre descrições e estimativas de esforço.

A.2.3 Análise de Legibilidade

A análise de legibilidade tem como objetivo avaliar a clareza e a complexidade linguística da descrição textual da *User Story*, permitindo verificar se o nível de compreensão.

Indicadores Gerais

A aplicação exibe para o usuário, depois de informado o texto da *User Story*, a quantidade de palavras, a quantidade de sentenças e a quantidade de sílabas, com a classificação dos índices. Esses valores podem indicar maior ou menor densidade lexical; em geral, densidade lexical elevada reduz a fluidez da leitura, enquanto densidade baixa tende a facilitá-la.

Índices de Legibilidade

Entre os principais índices avaliados, destacam-se:

- Flesch Reading Ease: Por exemplo se determinado texto tiver um valor de *-3,9*, ele será classificado como Muito Difícil, indicando que o texto é adequado apenas para leitores com nível de pós-graduação. Este índice não especifica uma área de formação (como Computação, Direito, Medicina etc.). O que ele indica é o nível geral de escolaridade necessário para compreender o texto. Quando o resultado é muito baixo, por exemplo 3,9, a classificação “Muito Difícil” significa: Apenas leitores com nível de pós-graduação conseguem ler com fluidez. Ou seja, espera-se que o leitor tenha alto domínio da língua e experiência em leitura de textos acadêmicos complexos. Este índice só mede o comprimento das frases, complexidade das palavras (sílabas) e a fluência textual. Ele não analisa conteúdo técnico, apenas a dificuldade linguística. A faixa completa do Flesch Reading Ease e a semântica correspondente, incluindo o nível educacional típico associado a cada intervalo é apresentado no

Capítulo 2 na seção 2.4.3.

- Flesch-Kincaid Grade Level: Por exemplo se determinado texto tiver um valor de *19,3*, correspondendo também a um nível de pós-graduação. A faixa completa do Flesch-Kincaid Grade Level e a semântica correspondente, incluindo o nível educacional típico associado a cada intervalo é apresentado no Capítulo 2 na seção 2.4.6.
- Gunning Fog Index: Por exemplo se determinado texto tiver um valor de 23,3, isto requer cerca de 23 anos de escolaridade formal para plena compreensão. A faixa completa do Gunning Fog Index e a semântica correspondente, incluindo o nível educacional típico associado a cada intervalo é apresentado no Capítulo 2 na seção 2.4.1.
- Coleman-Liau Index: Por exemplo se determinado texto tiver um valor de 10,4 – baseado em comprimento médio de palavras e sentenças. A faixa completa do Coleman-Liau Index e a semântica correspondente, incluindo o nível educacional típico associado a cada intervalo é apresentado no Capítulo 2 na seção 2.4.4.
- Automated Readability Index (ARI): Por exemplo se determinado texto tiver um valor de 11,6 – indica que o texto é adequado para leitores experientes. A faixa completa do Automated Readability Index e a semântica correspondente, incluindo o nível educacional típico associado a cada intervalo é apresentado no Capítulo 2 na seção 2.4.5.
- Dale-Chall Score: Por exemplo se determinado texto tiver um valor de 11,9 – baseado em vocabulário familiar, sinalizando o uso de muitos termos técnicos. A faixa completa do Dale-Chall Score e a semântica correspondente, incluindo o nível educacional típico associado a cada intervalo é apresentado no Capítulo 2 na seção 2.4.8.
- Linsear Write: Por exemplo se determinado texto tiver um valor de 19,0 – confirma a alta complexidade sintática e lexical. A faixa completa do Linsear Write e a semântica correspondente, incluindo o nível educacional típico associado a cada intervalo é apresentado no Capítulo 2 na seção 2.4.7.

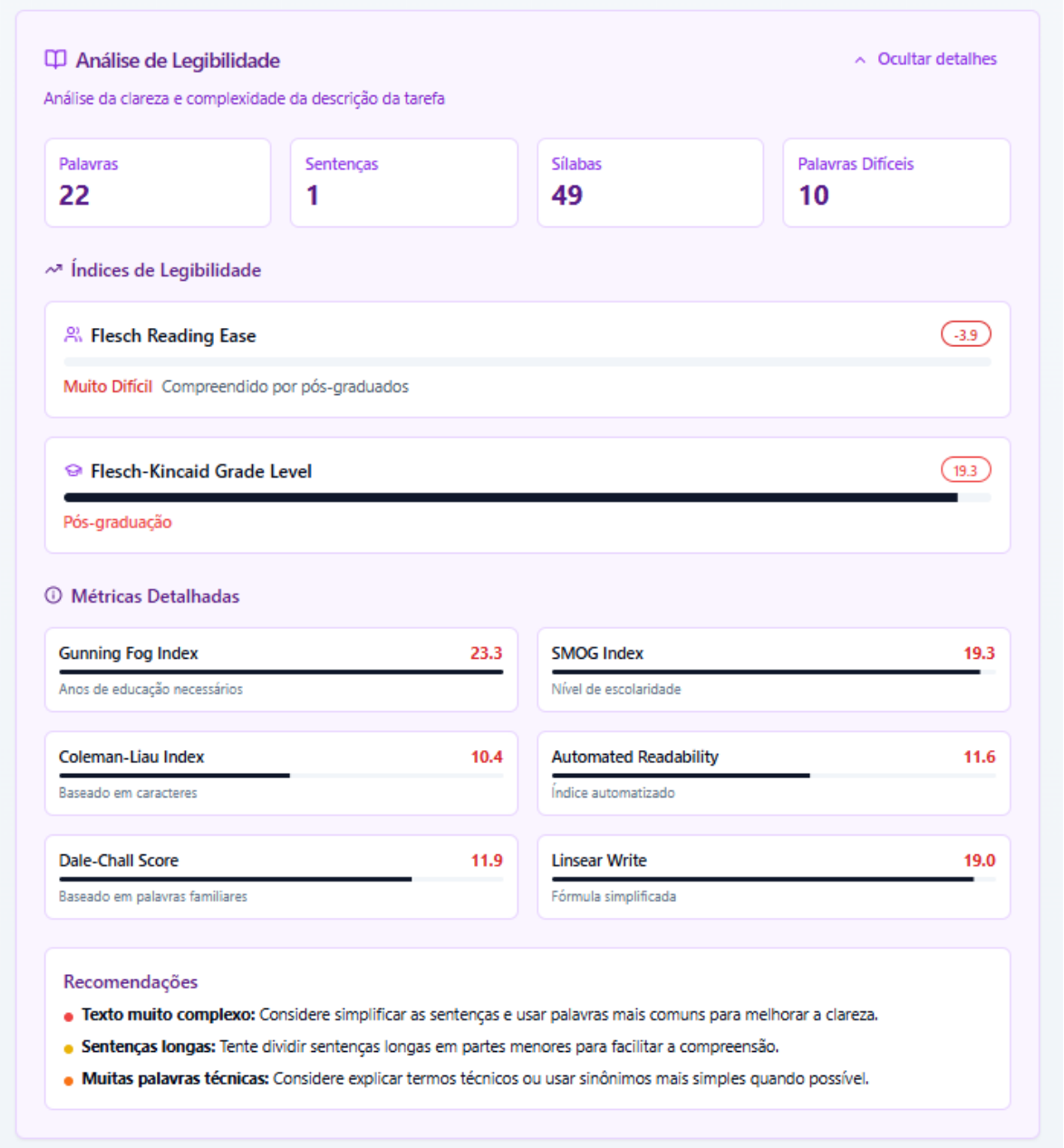
Recomendações de Simplificação

Com base nesses índices, podem ser sugeridas, por exemplo, as seguintes melhorias:

- Simplificação textual: Reduzir o número de palavras complexas e substituí-las por sinônimos de uso mais comum.
- Segmentação: Dividir a sentença longa em partes menores, favorecendo a compreensão progressiva.
- Clarificação técnica: Explicar ou substituir termos especializados que possam dificultar o entendimento para leitores não especialistas.

Essas ações contribuem para melhorar os valores dos índices, tornando o texto mais acessível e adequado sem comprometer o rigor técnico.

Figura A.10: Analise de Legibilidade



A.3 Perguntas Frequentes

Qual método de estimativa é mais preciso? Depende do contexto. A estimativa baseada em regras é consistente e explicável. O modelo *BERT fine-tuned* (Neo LLM Predictor) é especializado em *story points*. A IA oferece uma análise mais sofisticada. O aprendizado de máquina com aprendizagem profunda se

torna mais preciso com o tempo. Recomendação: utilize múltiplos métodos e compare os resultados.

O que é o modelo BERT fine-tuned? É um modelo *DistilBERT* que foi especificamente treinado pelo autor da tese (*fine-tuned*) (Neo LLM Predictor) com dados reais de *story points* de projetos de software do JIRA. Ele combina título e descrição para realizar uma análise contextual especializada, retornando tanto a estimativa quanto a confiança da predição.

Quantas tarefas preciso para treinar o ML? É necessário um mínimo de 5 tarefas para iniciar o treinamento, mas recomenda-se utilizar 20 ou mais para alcançar boa precisão. O modelo melhora continuamente com cada nova tarefa salva.

Os dados ficam seguros? Sim! Todos os dados são armazenados localmente no seu navegador. O modelo de ML também é salvo localmente. Para o *BERT* e a IA (Groq/Grok), apenas o texto da tarefa é enviado para processamento, sem armazenamento permanente.

E se a IA ou o BERT não funcionarem? A estimativa baseada em regras sempre funciona como *fallback*. O sistema foi projetado para ser resiliente — você sempre terá uma estimativa disponível, mesmo que os serviços externos estejam indisponíveis. Os serviços externos que a aplicação consulta são as APIs dos Modelos LLM Llama 3.1 disponível no GROQ, e o LLM Grok da xAI.

Posso personalizar as palavras-chave? Sim! Na aba **Configurações** é possível adicionar, remover e modificar todas as palavras-chave utilizadas na estimativa baseada em regras, adaptando o sistema ao seu domínio específico.

Como interpretar a análise de legibilidade? A análise de legibilidade ajuda a identificar se a descrição da tarefa está clara. Textos mais legíveis geralmente resultam em estimativas mais precisas e melhor entendimento por parte da equipe.

Qual a diferença entre BERT e IA generativa? O *BERT* é especializado em classificação de *story points*, pois foi treinado especificamente para essa tarefa. A IA generativa (Groq/Grok) realiza uma análise mais ampla e contextual, mas não foi treinada especificamente para *story points*. Dica: use ambos para comparação e maior precisão.

Dica 1: A ferramenta é mais eficaz quando utilizada de forma consistente. Comece simples, vá adicionando complexidade gradualmente na declaração em linguagem natural da User Story e sempre compare as estimativas com a realidade para melhorar o sistema.

Dica 2: *Story points* são relativos à sua equipe. Use esta ferramenta como apoio, não como substituto do julgamento humano. O modelo *BERT fine-tuned* oferece uma perspectiva especializada, mas a decisão final deve sempre considerar o contexto específico do seu projeto.

Outras telas do novo estimador de Story Points aparecem nas Figuras A.11 até A.18.

Como trabalho futuro sugere-se novos experimentos empíricos com usuários e cenários variados de User Story: por exemplo, User Stories simples, medianas e complexas, e comparações com estimativas.

Experimentos com User Stories com textos bem escritos e mal escritos será apresentado no Apêndice B, a seguir.

Figura A.11: Tela que exibe as estimativas das *User Story* importadas e das estimadas realizadas pelo aplicativo.

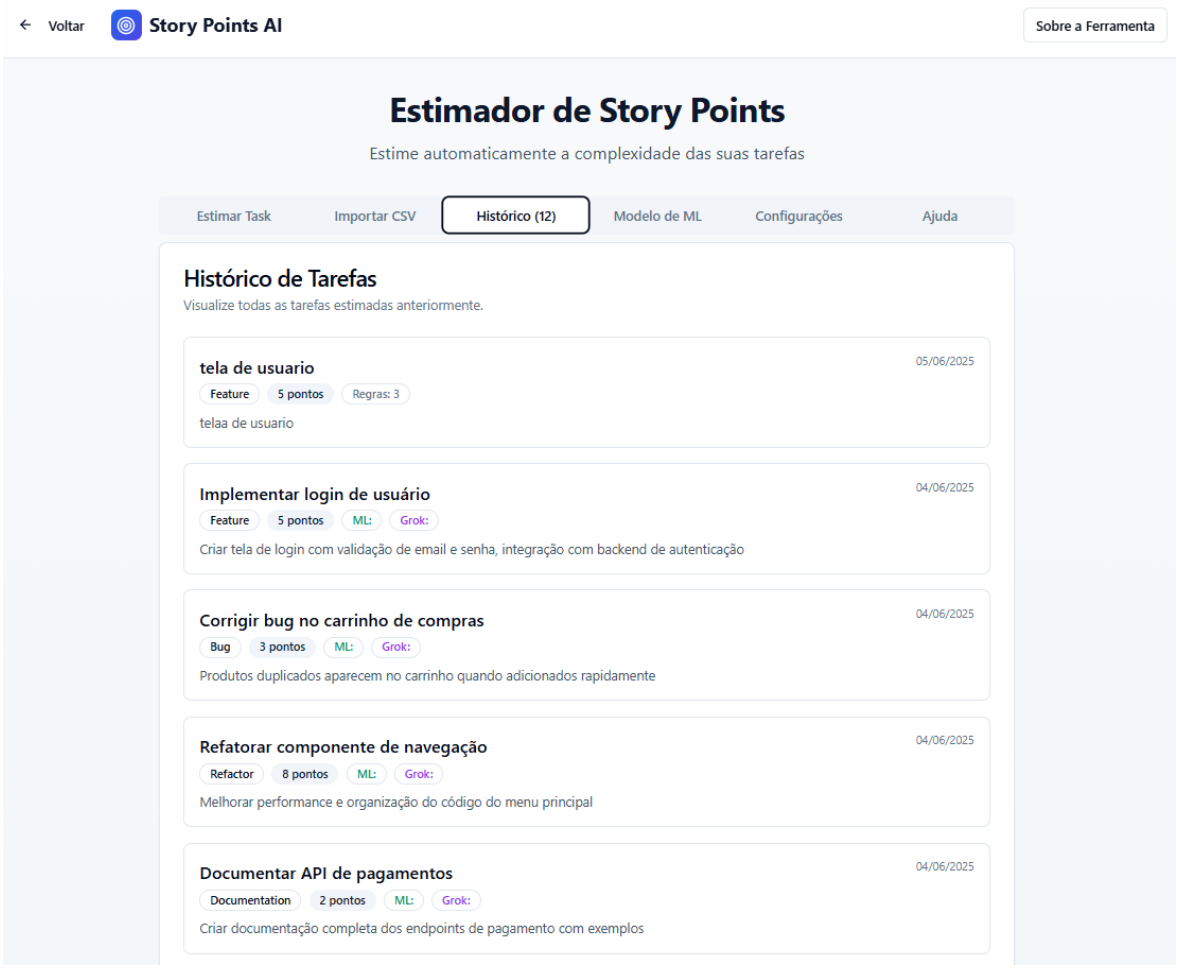


Figura A.12: Configurações das palavras chave de complexidade da estimativa com fórmulas Manuais para o (i) Modelo Baseado em Regras

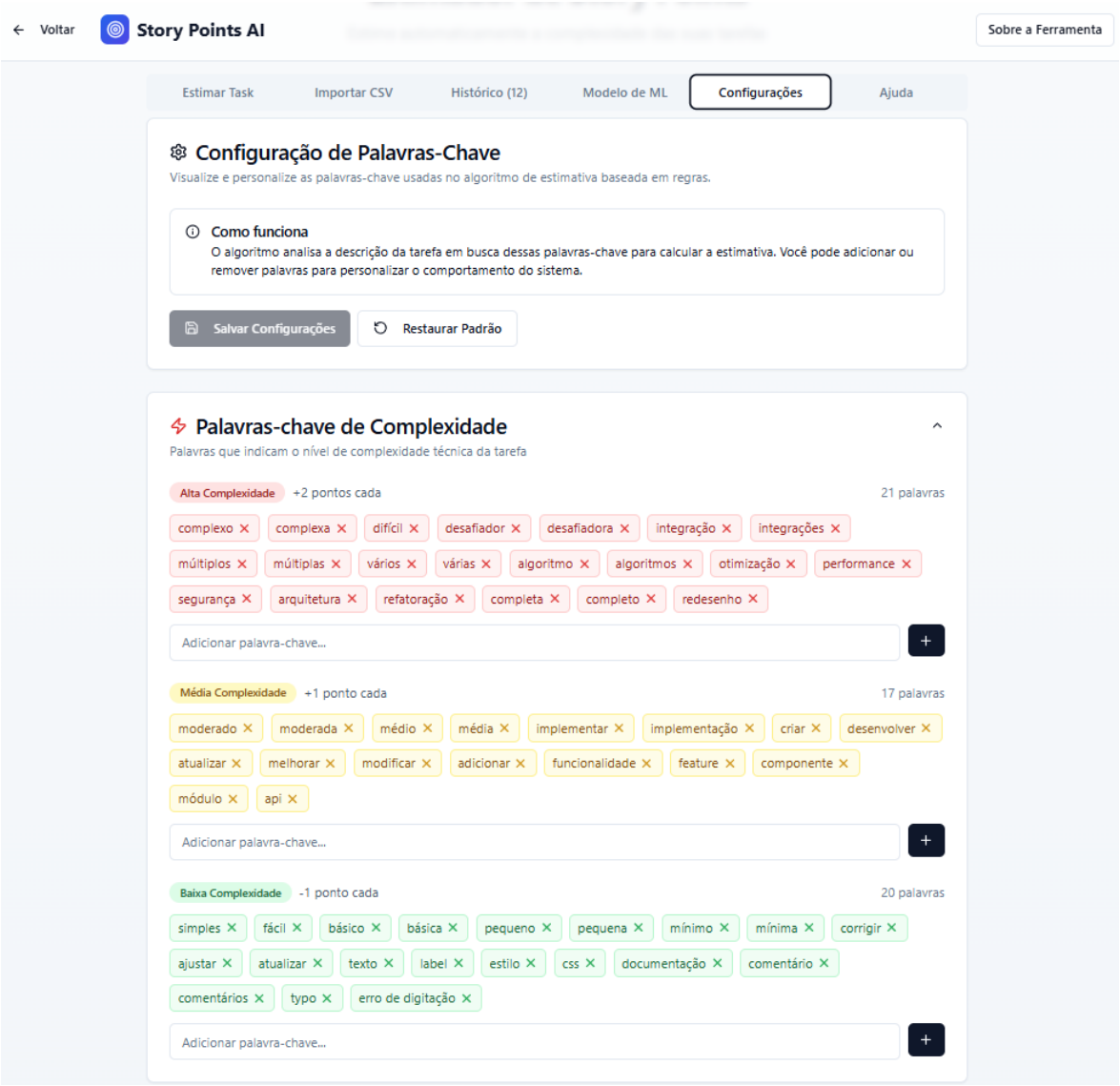


Figura A.13: Configurações das palavras chave de escopo da estimativa com fórmulas Manuais para o (i) Modelo Baseado em Regras.

Figura A.14: Configurações das palavras chave de Depêndencia da estimativa com fórmulas Manuais para o (i) Modelo Baseado em Regras

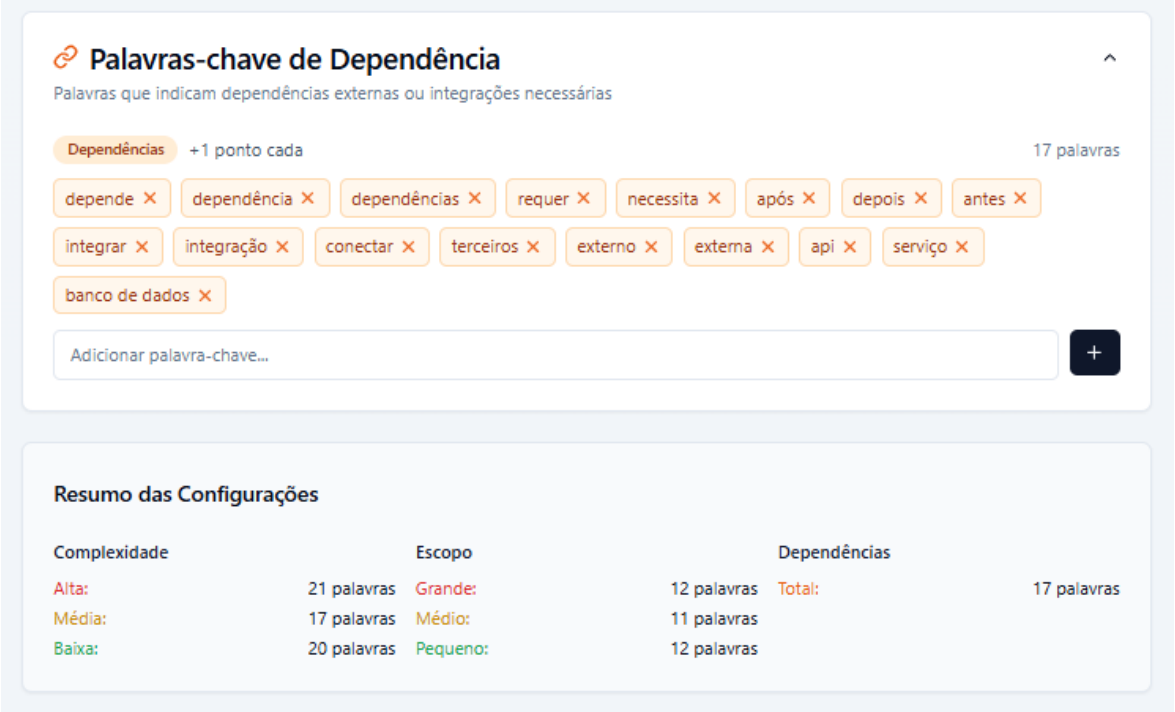


Figura A.15: Explicação dos Métodos disponíveis - Parte 1.

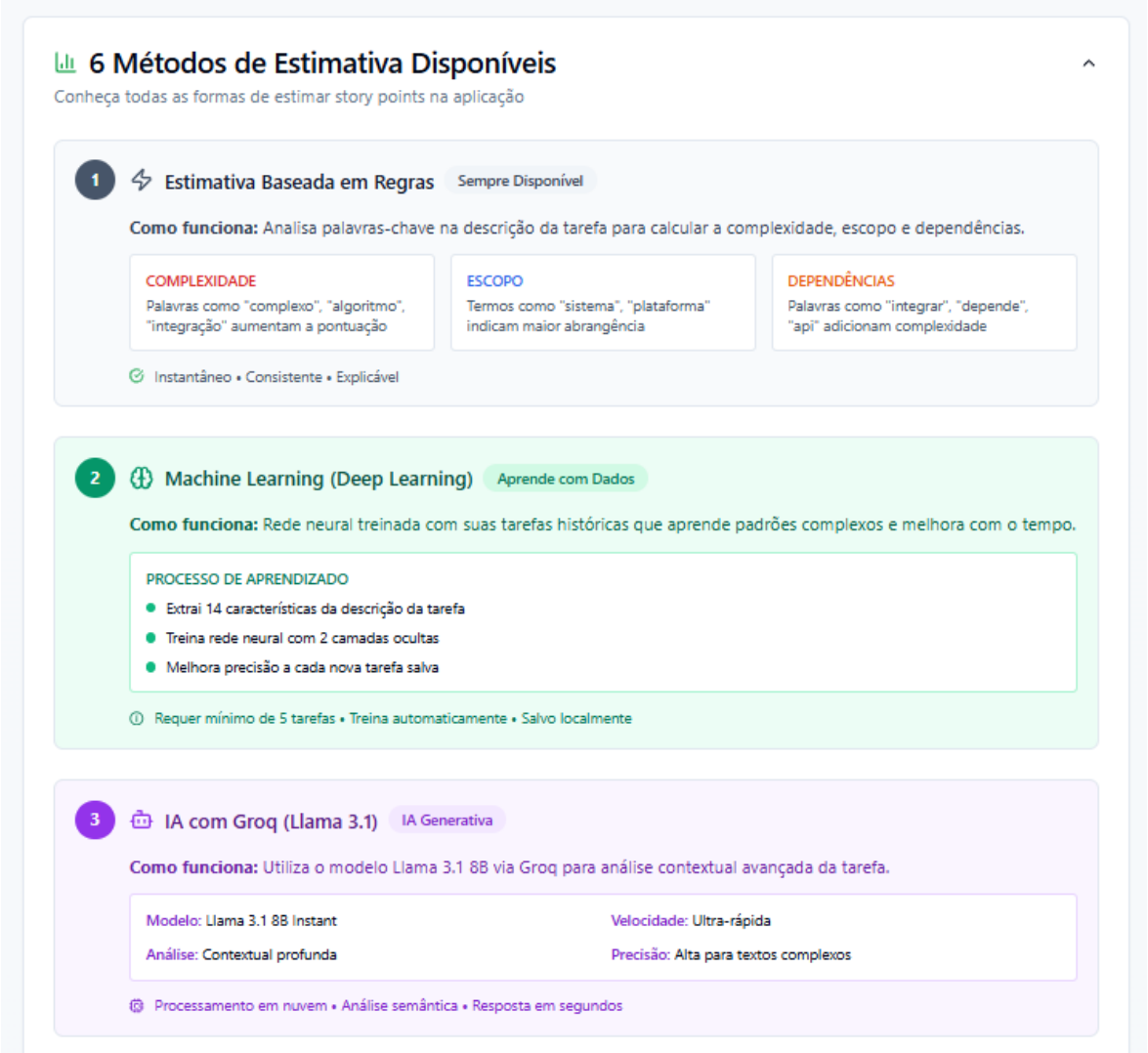


Figura A.16: Explicação dos Métodos disponíveis - Parte 2.

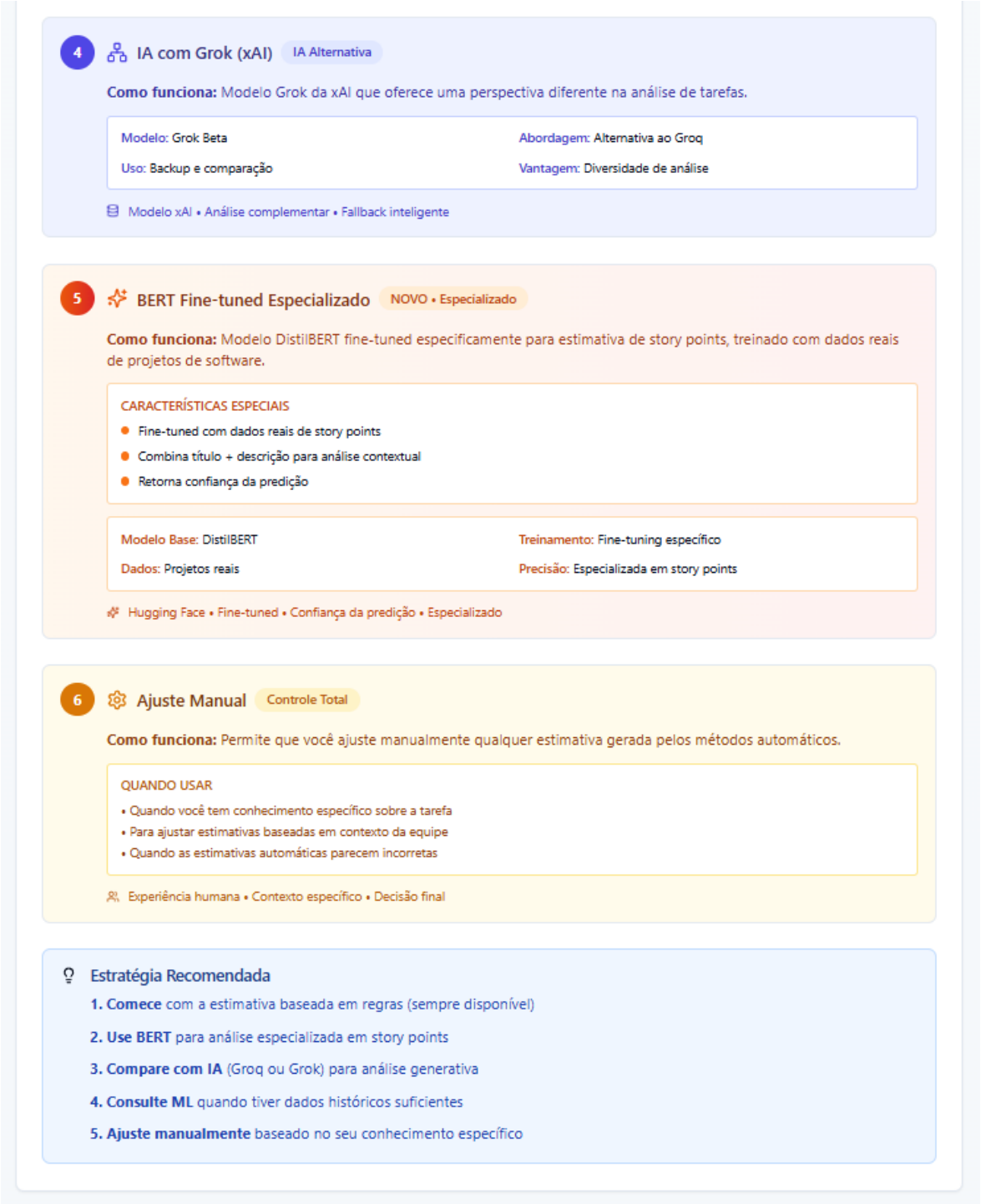
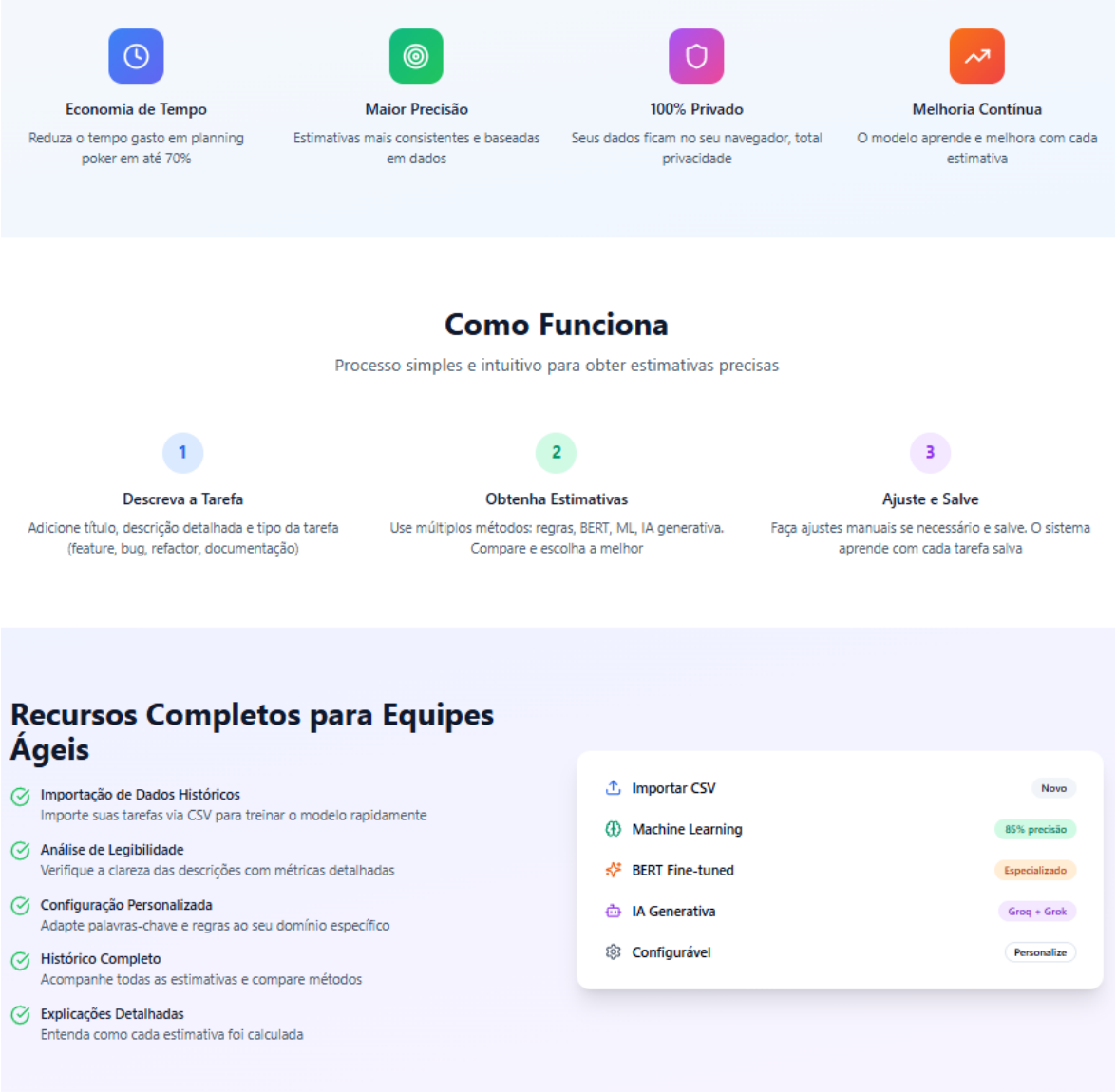


Figura A.17: Importação e detalhes de configuração avançados.

Figura A.18: Recursos disponíveis na aplicação. (Landing Page)



A.4 Disponibilidade dos artefatos

- Uma versão on-line está disponível em `https://neosp.vercel.app/`.
- O código fonte está disponível em `https://github.com/giseldo/estimador_de_story_points`.

Apêndice B

Melhoria do texto nas *User Stories* e o impacto nas estimativas Cross-Project

B.1 Introdução

Com o objetivo de verificar se a reescrita do texto da *User Story* influencia a mensuração automática de esforço, foram submetidos dois conjuntos de descrições ao modelo de estimativa *Cross-Project*: uma versão de uma *User Story* propositalmente ambígua e mal estruturada e uma versão ajustada. Essa comparação permite observar a sensibilidade do modelo de mensuração de esforço em relação às variações linguísticas e semânticas das descrições.

B.2 Metodologia

Cada par de *User Stories* (ruim e aprimorada) foi submetido ao modelo de mensuração de esforço *Cross-Project* , sendo registrada a estimativa de *story points* em ambos os casos. Espera-se que *User Stories* mais claras, objetivas e delimitadas apresentem estimativas menores e mais consistentes. Em contraste, *User Stories* confusas ou de escopo indefinido tendem a gerar valores superestimados. Veja

na Figura B.1 e na Figura B.2 como o experimento foi conduzido utilizando o *User Story* Tutor.

Figura B.1: Uma estimativa com texto ruim (parte 1).

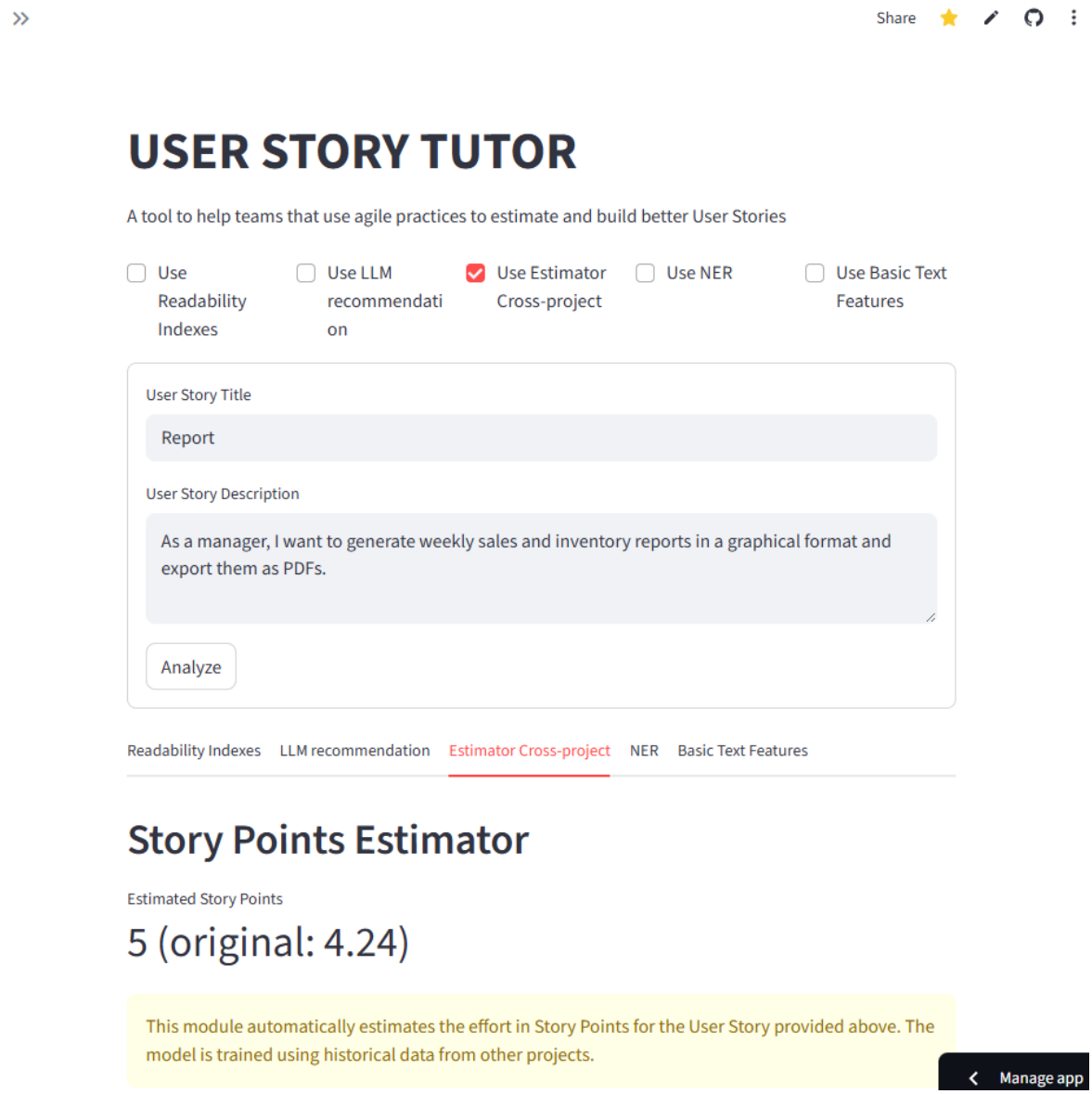
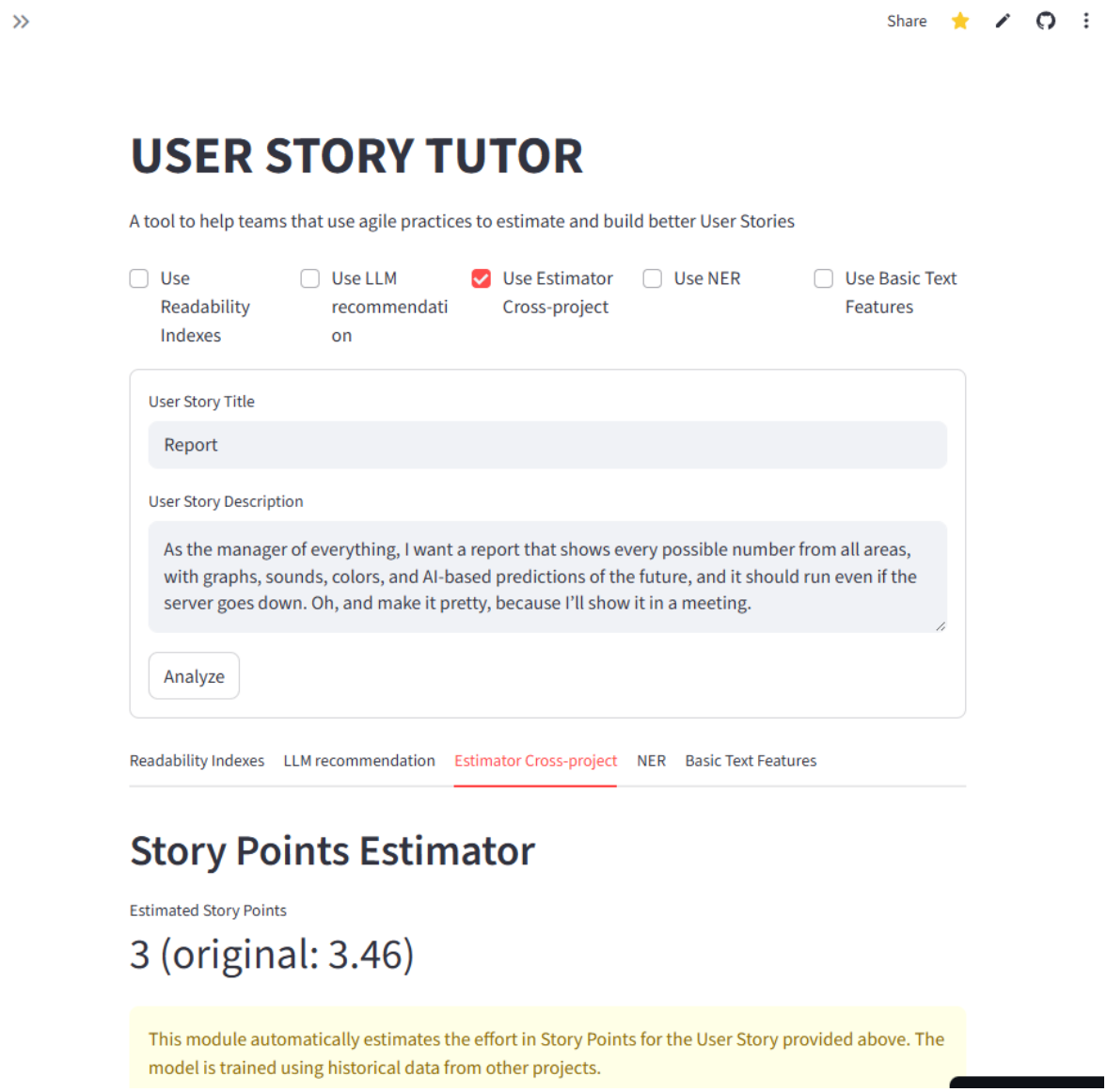


Figura B.2: Uma estimativa com texto melhorado (parte 2).



A Tabela B.1 contem textos de dez *User Stories*. Na coluna "texto ruim", há um texto insatisfatório para uma *User Story*, e na coluna "texto melhorado", existe uma versão aprimorada da *User Story* (menos confusa).

B.3 Resultados

A Tabela B.2 apresenta os valores simulados de estimativa de esforço (*story points*) atribuídos às dez *User Stories* analisadas. Nota-se uma diferença consistente nas estimativas após a reescrita das descrições.

Tabela B.1: Texto das *User Stories* ruins e melhoradas

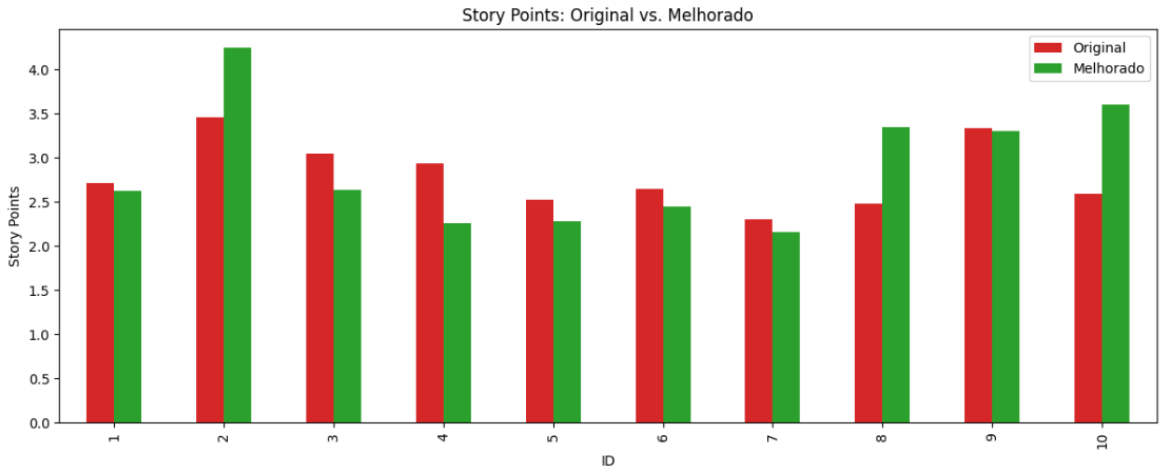
ID	Texto ruim	Texto melhorado
1	As someone who uses the system (I think), I want to log in or maybe not log in, depending on whether I'm already inside, because sometimes the system logs me out by itself and it makes me mad. It would be nice if it remembered who I am without asking for my password too often, because I forget it. Also, maybe an 'anonymous' mode but with full admin access.	As an authenticated user, I want to log in using my email and password and sign out in 30 minutes of inactivity to avoid repeated logins.
2	As the manager of everything, I want a report that shows every possible number from all areas, with graphs, sounds, colors, and AI-based predictions of the future, and it should run even if the server goes down. Oh, and make it pretty, because I'll show it in a meeting.	As a manager, I want to generate weekly sales and inventory reports in a graphical format and export them as PDFs.
3	As a user, I want to search for anything, even if I don't know the name — like 'that blue product I saw last week' — and the system should understand my feelings and show me something similar, but only if I'm in a good mood.	As a user, I want to search for products by name or category with automatic suggestions of category based on the text I type.
4	I want to register but without filling out anything, just one click, and if it fails, it should keep trying until it works. Also, if I already have an account, it should create another one anyway so there's no conflict.	As a new user, I want to fill in my name, email, and password and receive a confirmation email to activate my account.
5	As the supreme admin, I want a dashboard that shows everything about everyone, in real time, with automatic audio alerts whenever someone even *thinks* of clicking something wrong. And it must work offline, of course.	As an admin, I want a dashboard with user activity and system error statistics, updated hourly.
6	I want to be notified about everything, always — even things I don't care about — because I'd rather know than not know. The system can send emails, SMS, faxes, or holograms — whatever works.	As a user, I want to receive email notifications when my order is shipped or canceled.
7	I want to pay with credit card, bank slip, Pix, crypto, lunch voucher, a smile, or positive energy. If I have no balance, the system should trust me to pay later because I'm honest.	As a customer, I want to pay using a credit card, bank slip, or Pix during checkout.
8	I want the system to be 100% secure but also easy to access, and I should be able to see other people's passwords just to make sure they're safe. Maybe add a 'hack system' button to check for vulnerabilities.	As a user, I want the system to use secure authentication and password encryption.
9	I want to integrate the system with every service on Earth, including the ones that don't exist yet, like 'ChatGPT 12' and the 'Intergalactic Bank.' Ideally, it should all run in real time and be free.	As an admin, I want to integrate the system with the Google Calendar API to synchronize events automatically.
10	want the system to be instantaneous — faster than human thought — even on dial-up Internet. If it takes longer than a blink, it's unacceptable.	As a user, I want pages to load in under two seconds on a 4G connection.

Tabela B.2: Comparação entre estimativas para versões ruins e melhoradas de *User Stories*

ID	Título da US	Versão Ruim	Versão Melhorada
1	Chaotic Login	13	3
2	Divine Report	20	5
3	Transcendental Search	16	4
4	Nonsensical Registration	11	3
5	Omnipotent Dashboard	21	6
6	Delusional Notifications	15	4
7	Supernatural Payment	18	5
8	Schizophrenic Security	19	5
9	Megalomaniac Integration	22	6
10	Impossible Performance	17	4

A Figura B.3 ilustra a diferença entre as estimativas das versões ruins e melhoradas por meio de um gráfico

Figura B.3: Resultados



Observa-se que todas as *User Stories* melhoradas apresentaram diferenças no esforço estimado. Esse resultado reforça a hipótese de que a qualidade textual das *User Stories* impacta diretamente o desempenho do modelo de estimativa de esforço, tanto pela redução da ambiguidade quanto pela delimitação mais precisa do escopo funcional.

Os resultados indicam que o modelo proposto é sensível à qualidade da formulação textual das *User Stories*, o que sugere uma relação negativa entre User Stories mais claras e menor estimativa de esforço. Neste sentido, o modelo é consitente.

para a confiabilidade das estimativas automáticas de esforço. Sendo assim, o uso do LLM é um fator de melhoria no texto da user story.

B.4 Disponibilidade dos artefatos

- O código fonte está disponível em `https://github.com/giseldo/userstory` no diretório “experiment”.

Apêndice C

Revisão Sistemática

O maior inimigo do conhecimento não é a ignorância, é a ilusão do conhecimento.

Stephen Hawking

Resumo

CONTEXTO: O Processamento de Linguagem Natural (PLN) aplicado à estimativa de esforço no desenvolvimento de software é uma área de pesquisa e desenvolvimento que visa aplicar técnicas para estimar o esforço dos projetos a partir da análise textual de artefatos disponíveis.

OBJETIVO: O objetivo deste apêndice é identificar as técnicas, os artefatos textuais utilizados como preditores e as métricas utilizadas na estimativa de esforço com o uso de PLN.

MÉTODO: Foi realizada uma revisão sistemática em cinco Bases de Busca e foram selecionados doze estudos primários para a extração dos dados.

RESULTADOS: Foi identificado que as técnicas *Term Frequency–Inverse Document Frequency* (TF-IDF) com *Support Vector Machine* (SVM) são as mais populares; um dos artefatos de entrada mais utilizados é o texto da User Story; já as métricas de avaliação mais utilizadas foram o MAE e o MMRE.

CONCLUSÃO: No geral, embora a área esteja em ascensão, ainda há necessidade de estudos mais empíricos que possibilitem novas contribuições científicas.

C.1 Introdução

Uma *User Story* é um artefato textual; por isso, foi realizada uma busca pelas técnicas de Processamento de Linguagem Natural (PLN) existentes na literatura especializada neste domínio. Neste apêndice, apresenta-se a metodologia da revisão, os resultados da extração e a análise dos resultados.

C.2 Metodologia

A metodologia escolhida foi uma revisão sistemática da literatura, seguindo as orientações de [71]. Foi utilizado o software Parsif.al [1] para registrar o protocolo da pesquisa e manter o registro das etapas realizadas.

Com base nas questões de pesquisa, foi proposto o termo de busca com o auxílio do PICOC (População, Intervenção, Comparação, Resultado e Contexto) [11]. A versão final do PIPOC é apresentada na tabela C.1.

Tabela C.1: PICOC da Revisão Sistemática

Aspecto	**Valor**
População (P)	Estimativas de Esforço
Intervenção (I)	Técnicas de PLN
Comparação (C)	Outras Revisões da Literatura
Resultados (O)	Pesquisas onde o PLN é utilizado para estimar esforço em desenvolvimento de software.
Contexto (C)	Projetos de desenvolvimento de software

Foi utilizado como termo principal “Estimativas de Esforço” para representar o contexto da pesquisa e “Processamento de Linguagem Natural” para representar a intervenção nesse contexto; usou-se o Boolean “OR” para incorporar sinônimos e ortografias alternativas em cada conjunto de termos, e “AND” para vincular os dois conjuntos. O conjunto geral dos termos de pesquisa é apresentado a seguir.

(“effort estimation” OR “software development effort estimation”)
AND (“natural language processing” OR “NLP”)

[1] https://parsif.al/

As 5 bases de busca utilizadas nessa revisão são apresentadas na tabela C.2, junto com o termo de busca de cada uma delas. Essas bases foram escolhidas por indexarem algumas das principais revistas e conferências relacionadas à engenharia de software [18].

Tabela C.2: Bases e termo de busca

Nome	**URL**	**Termo de busca**
ACM Digital Library	`https://dl.acm.org/`	[Abstract: estimation or estimate] AND [Abstract: effort] AND [Abstract: natural] AND [Abstract: language]
IEEE Digital Library	`https://ieeexplore.ieee.org/`	natural language processing estimation effort
Science Direct	`https://www.sciencedirect.com/`	Title, abstract, keywords: (’natural language’) AND (’estimation effort’ OR ’effort estimation’ OR ’estimate effort’)
Scopus	`https://www.scopus.com/`	TITLE-ABS-KEY (natural AND language AND estimation AND effort AND software)
Web of Science	`http://webofscience.com/`	(“Effort Estimation” OR “Estimation Effort”) AND “Natural Language Processing”

Os estudos encontrados foram revisados de acordo com critérios de inclusão e exclusão previamente estabelecidos para orientar a etapa de seleção dos estudos. Os critérios são apresentados na tabela C.3.

Tabela C.3: Critérios de inclusão e exclusão

Tipo	**Descrição do critério inclusão e exclusão**
Inclusão	Estudos que aplicam técnicas de PLN na Estimativa de Esforço
Exclusão	Estudos não disponíveis ou incompletos
Exclusão	Estudos duplicados
Exclusão	Estudos que não são relevantes ou de baixa qualidade
Exclusão	Livros
Exclusão	Artigos que não sejam escritos em inglês.

Foram adotados também, critérios para classificação dos artigos quanto à sua qualidade, em que a nota de cada critério de qualidade varia entre 0, 0,5 e 1. Caso o artigo atendesse completamente ao critério estabelecido, receberia a nota máxima 1; caso o artigo atendesse parcialmente ao critério estabelecido, receberia índice

de qualidade 0,5 e, caso o artigo não atendesse ao critério estabelecido, receberia a nota mínima do índice de qualidade, ou seja, 0. Para essa pesquisa, foram utilizados os critérios de qualidade apresentados na tabela C.4. Por exemplo, se o artigo atingisse nota máxima em todos os critérios (c1 até c5), sua nota seria 5, pois foram avaliados 5 critérios.

Tabela C.4: Critérios de qualidade avaliados

ID	Descrição do critério de qualidade
c1	Os objetivos da pesquisa estão claramente definidos no estudo?
c2	As técnicas aplicadas estão claramente definidas no estudo?
c3	O uso do conjunto de dados está descrito claramente?
c4	Os pesquisadores dão detalhes suficientes para a avaliação dos métodos?
c5	Os resultados são descritos e discutidos claramente?

C.3 Resultados

No total, 118 artigos foram selecionados inicialmente sem a análise dos critérios. Destes, 24 estavam duplicados e foram removidos. Após a leitura do título e do resumo, 71 foram removidos, pois não atenderam aos critérios de inclusão. Foram selecionados então 23 artigos para leitura completa. Depois da leitura, 10 artigos foram removidos e 2 também foram excluídos por ainda se tratarem do mesmo experimento — mesmo com nomes diferentes. Ao final, 12 artigos foram aceitos e selecionados para a extração dos dados. Um fluxograma do protocolo da revisão é apresentado na Figura C.1.

C.3.1 Artigos Selecionados

Os artigos aceitos para extração dos dados, que passaram pelo critério de inclusão, são apresentados na tabela C.5, junto com seus critérios de qualidade. A figura C.2 apresenta o tipo de veículo (se conferência ou journal) em que o artigo selecionado foi publicado. A maioria dos artigos aceitos foi publicada em conferências. Os parágrafos a seguir apresentam uma análise resumida de cada um dos artigos aceitos.

Figura C.1: Fluxograma do protocolo da revisão.

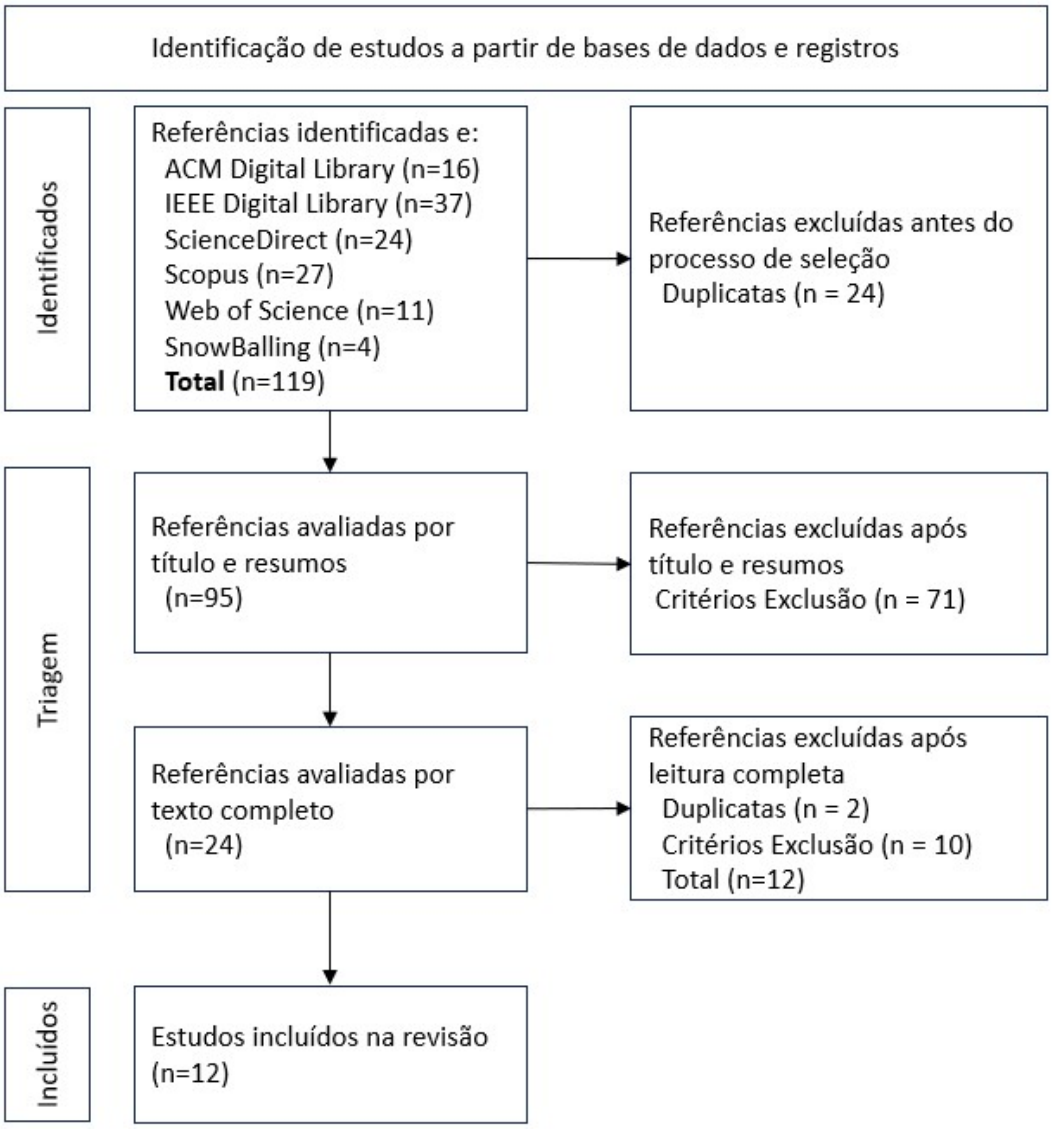
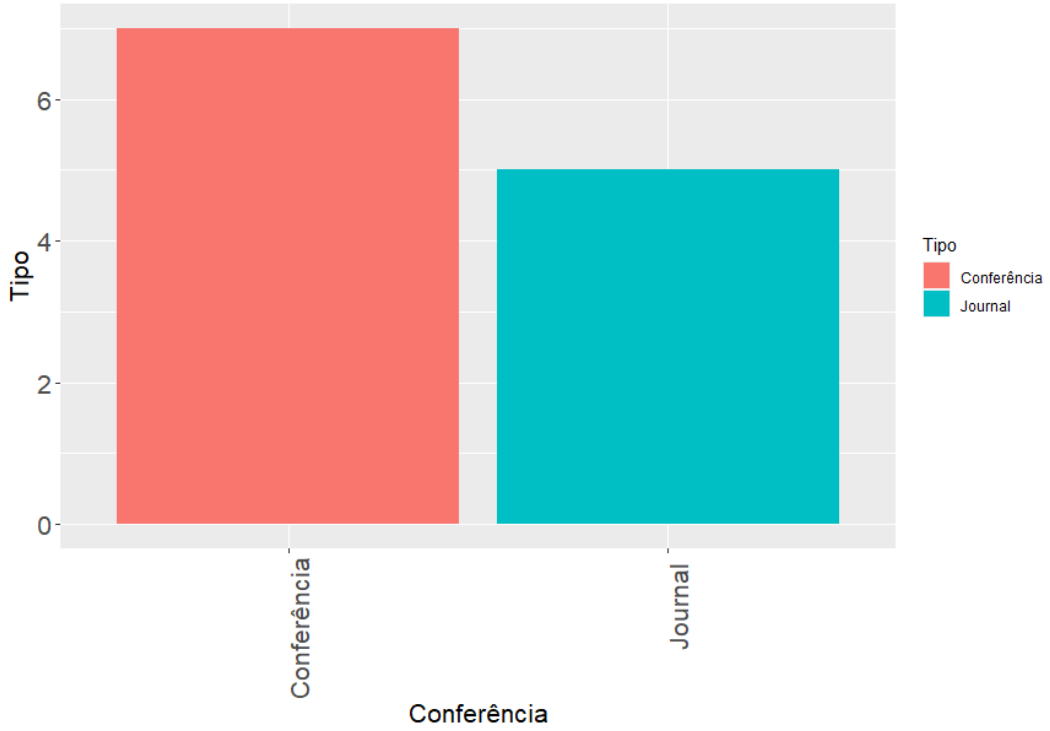


Tabela C.5: Avaliação dos Critérios de Qualidade

Título	**Conferência ou Journal**	**Ano**	**Qual.**	**Cit.**
#1 An NLP Approach to Estimating Effort in a Work Environment [31]	International Conference on Software, Telecommunications and Computer Networks (SoftCOM)	2020	3,0	4
#2 Effort estimation approach through extracting use cases via informal requirement specifications [106]	Applied Sciences (Switzerland) v.10, n.9	2020	4,0	13
#3 Customer requests matter: Early stage software effort estimation using k-grams [41]	Proceedings of the 35th Annual ACM Symposium on Applied Computing	2020	4,5	2
#4 A Deep Learning Model for Estimating *Story Points* [23]	IEEE Transactions on Software Engineering, v.45, n.7	2019	5,0	194
#5 Effort Estimation using Bayesian Networks for Agile Development [119]	2nd International Conference on Computer Applications Information Security (ICCAIS)	2019	4,0	9
#6 Using developers' features to estimate *Story Points* [130]	Proceedings of the 2018 International Conference on Software and System Process	2018	5,0	44
#7 Effort Estimation via Text Classification and Autoencoders [135]	2018 International Joint Conference on Neural Networks (IJCNN)	2018	5,0	21
#8 Natural Language Processing and Machine Learning Methods for Software Development Effort Estimation [62]	Studies in Informatics and Control, v.26, n.2	2016	4,5	13
#9 Estimating *Story Points* from issue reports [111]	Proceedings of the the 12th international conference on predictive models and data analytics in software engineering	2016	5,0	68
#10 Functional size approximation based on usecase names [100]	Information and Software Technology, v.80	2016	4,0	20
#11 ESSE: An early software size estimation method based on auto-extracted requirements features [173]	Proceedings of the 8th AsiaPacific Symposium on Internetware	2016	4,0	9
#12 Approximation of COSMIC functional size to support early effort estimation in Agile [59]	Data & Knowledge Engineering, v.85	2013	4,5	70

Onde: **Qual** é a qualidade auferida pelo autor desta tese; e textbfCit é o número de citações que o artigo.

Figura C.2: Tipo de veículo onde o artigo foi publicado.



#1 [31] (An NLP Approach to Estimating Effort in a Work Environment): Dan et al. [31] propõem um sistema preditor de estimativa de esforço que prevê homem-hora e usa como preditor o texto da User Story. Além do texto, eles utilizam também algumas outras variáveis preditoras, tais como o tempo de experiência do desenvolvedor que trabalhará nessa tarefa, o cliente e o tipo de projeto. Eles sugerem um preditor que utiliza uma *deep neural network* em uma arquitetura sequencial simples. O modelo tem uma camada de *embedding* pré-treinada para mapear o texto em vetor; em seguida, o texto é reduzido para no máximo 20 palavras. A próxima camada é uma "masking layer", uma camada LSTM com a função de ativação ReLU e uma camada de Dropout. Eles utilizaram o MMRE para comparar o desempenho do estimador. Segundo os autores, o preditor obteve melhores resultados do que os sistemas de estimativa clássicos, sendo equivalente aos estimadores humanos; porém, não foram encontradas evidências no texto para apoiar essa afirmação.

#2 (Effort estimation approach through extracting use cases via informal requirement specifications) [106]: Park and Kim [106] propõem um preditor de Use Case Point extraindo casos de uso dos requisitos textuais utilizando análise semântica. O

Use Case Point é outra medida utilizada em times ágeis, além dos Story Points. O modelo identifica verbos-chave nas sentenças do texto em linguagem natural e reconhece os argumentos dessa sentença. Em seguida, o modelo extrai um caso de uso por meio de um mecanismo de modelagem visual recursivo, até que não haja mais conexões. Finalmente, o esforço do software é calculado com base neste caso de uso extraído. Não foi feita nenhuma comparação com outras abordagens. Sendo avaliada com exemplos de descrição dos requisitos de um sistema de correio.

#3 (Customer requests matter: Early stage software effort estimation using k-grams) [41]: Eren et al. [41] propõem um modelo que prevê esforço em homem-minutos utilizando a descrição da User Story. Além das descrições, outros dados foram utilizados, tais como o tipo de aplicativo, o tipo da requisição e o mês da solicitação. O modelo aplica uma heurística para extrair *k-grams* e combina recursos não textuais da solicitação para criar o conjunto final de atributos preditores. Foram construídos 2 (dois) modelos: um que pode ser utilizado por todos os clientes (chamado pelo autor de Modelo Unificado) e modelos específicos, treinados com as solicitações daquele cliente (Modelos Específicos). O Modelo Unificado alcançou 43% de valores Pred (25) e 51% de SA. Já os Modelos Específicos oscilaram entre 39%-58% Pred(25) e 35%-58% de SA. Para treino do modelo, foram utilizadas 1.648 solicitações de clientes da organização. Alguns algoritmos de Aprendizagem de Máquina também foram comparados (*Regression Trees, Analogy Based Estimation, Linear Regression and Random Forest*). O *Random Forest* foi o algoritmo que apresentou os melhores resultados.

#4 (A Deep Learning Model for Estimating Story Points) [23]: Choetkiertikul et al. [23] propõem um modelo preditivo de *Story Points* a partir da descrição da *User Story* que utiliza *deep-learning*, *long short memory* e *recurrent highway network*. O conjunto de dados utilizado tem 23.313 *User Stories* de 16 projetos de código aberto. As métricas de avaliação utilizadas foram MAE, MdAE e SA. O modelo é comparado com 3 *baselines* (Random Guessing, Mean e Median) e 6 alternativas de modelos (por exemplo: Doc2Vec, Random Forests). Esse estudo foi reproduzido por Tawosi em 2023 [150] que não corroborou esses resultados.

#5 (Effort Estimation using Bayesian Networks for Agile Development) [119]:

Ratke et al. [119] propõem um modelo que usa Naive Bayes para estimar o esforço em *Story Points* a partir da descrição da User Story. Primeiro, eles criaram uma tabela com palavras, a quantidade de repetição dessa palavra e a probabilidade de que uma *User Story* com determinado *Story Point* contenha essa quantidade. Para a previsão, o modelo calcula uma probabilidade condicional a partir da extração de palavras da User Story. São extraídos os verbos e substantivos da User Story; depois, é feita uma redução verbal (verbos no infinitivo) e uma busca por sinônimos; por fim, calcula-se a probabilidade condicional. A avaliação e teste do modelo em ambiente real utilizaram requisitos no formato de Desenvolvimento Orientado a Comportamento (Behavior Driven Development - BDD). A métrica reportada foi 85% de Acurácia.

#6 (Using developers' features to estimate story points) [130]:Scott and Pfahl [130] propõem um modelo que prevê o esforço em *Story Points* a partir da descrição da *User Story* e de 3 outras variáveis, a reputação, a carga de trabalho e o total da capacidade de trabalho do desenvolvedor. Eles criaram um modelo preditivo com atributos somente do texto da *User Story* e outro modelo com atributos do desenvolvedor e do texto. Eles fizeram uma comparação entre os dois modelos, além de uma comparação com os *baseline Random Guessing*, Média e Mediana. Os autores encontraram resultados semelhantes entre os dois modelos; o modelo com os atributos extraídos do desenvolvedor e do texto é melhor do que um modelo que possui apenas os atributos do texto. Para avaliar seus resultados, eles utilizaram a Acurácia, o MAE e o SA. A amostra foi de 15.155 *User Stories*, porém, depois do processo de limpeza, a quantidade diminuiu para 4.142. Eles usaram o mesmo conjunto de dados de Porru et al. (2016) [111]. O Modelo utiliza uma estratégia de classificação com o algoritmo SVM e, para extrair os atributos do texto, utilizou TF-IDF.

#7 (Effort Estimation via Text Classification and Autoencoders) [135]: Soares [135] propôs um modelo preditivo de *Story Points* extraindo atributos do texto da *User Story* com autoencoders e utilizando o algoritmo SVM. O modelo transforma um *bag-of-words* de alta dimensão em uma representação de texto em vetores de baixa dimensão. Os autores também realizaram a otimização dos parâmetros do

modelo com validação cruzada. Eles afirmam que os experimentos forneceram evidências de que os autoencodificadores conseguiram codificar documentos ruidosos e oferecer recursos informativos ao classificador. Foram apresentadas as métricas: precisão, sensibilidade e Medida-F. O conjunto de dados utilizado no treinamento foi de *User Stories* selecionadas de 6 projetos código aberto, totalizando 3.439 *User Stories*.

#8 (Natural Language Processing and Machine Learning Methods for Software Development Effort Estimation): Ionescu et al. [62] propõem um modelo preditivo de homem-minutos utilizando a extração de atributos do texto da *User Story* com TF-IDF e com o algoritmo SVM. Esse modelo aplica um pré-processamento na descrição da *User Story*, utilizando estratégias básicas de análise e extração de valores numéricos a partir do texto. Eles compararam sua abordagem com outro extrator de atributos do texto: o Doc2Vec. A vetorização básica do TF-IDF apresentou melhores resultados do que o Doc2Vec quando utilizada na regressão. Os dados utilizados foram de uma empresa real de desenvolvimento de software, com 9 projetos totalizando 4.402 *User Stories*. A métrica reportada foi o MMRE.

#9 (Estimating *Story Points* from issue reports) [111]: Porru et al. [111] propõem um preditor de estimativa de *Story Points* a partir da Descrição da *User Story* utilizando como extrator de atributos o TF-IDF. A amostra utilizou vários projetos retirados do Jira [9] totalizando 5.607 *User Stories*. O modelo junta o Título e a Descrição em uma nova coluna, chamada contexto, e separa do contexto o que é código fonte, do que é somente texto em linguagem natural. A conclusão dos autores é a de que a descrição da *User Story* carrega informações relevantes para a estimativa automática. Como trabalho futuro, sugeriram um plugin do Jira [9] que iria estimar automaticamente e marcar as palavras mais relevantes para a estimativa. Como métrica, utilizaram a acurácia e o MMRE. Suas conclusões são de que treinar o classificador com mais de 200 Tarefas é mais eficiente, sugerindo mais de 300 Tarefas para obter um MMRE maior do que 0,61. Como baseline, utilizaram o algoritmo ZeroR. Eles compararam os algoritmos SVM, KNN, *Decision Tree e Naïve Bayes*. O SVM foi o que obteve os melhores resultados do MMRE. Esse estudo foi reproduzido por Tawosi em 2023 [150].

#10 (Functional size approximation based on use-case name) [100]: Ochodek [100] propôs um modelo preditivo do tamanho funcional COSMIC a partir do texto do Título do caso de uso. Dada uma lista de nomes de casos de uso, o modelo aproxima o tamanho funcional COSMIC. O método utilizado segue duas etapas. Na primeira etapa, são processados os nomes dos casos de uso, expressos em uma linguagem natural e classificados em uma das treze categorias. Na segunda etapa, as informações sobre categorias e dados históricos são empregadas para aproximar o tamanho em Pontos de Função COSMIC. A precisão foi avaliada por meio de um procedimento de validação cruzada em um conjunto de dados de 437 Casos de Uso de 27 projetos de desenvolvimento de software.

#11 (ESSE: An early software size estimation method based on auto-extracted requirements features) [173]: Zhang et al. [173] propôs um modelo preditivo de esforço, porém, a medida do esforço não foi especificada. Ele utilizou regressão a partir dos requisitos em linguagem natural utilizando análise semântica. O modelo extrai 5 informações do texto (User, Action, Object, ActionProperty, Condition) utilizando *Entity* e extração de atributos e transforma o texto em uma matriz com o suporte do WordNet. Foram utilizados 508 requisitos textuais escritos em linguagem natural de 21 projetos de empresas reais. Foi utilizada a MMRE e o PRED (25) como métricas. Essa base de dados não foi disponibilizada no texto, o que compromete a reprodutibilidade do artigo.

#12 (Approximation of COSMIC functional size to support early effort estimation in Agile) [59]: Hussain et al. [59] propôs um modelo preditivo de classificação, com árvore de decisão, do tamanho funcional COSMIC a partir dos requisitos textuais utilizando análise sintática. O modelo utiliza o Stanford Parser para a extração das características (frequência dos substantivos, frequência dos parênteses, frequência dos verbos, frequência dos pronomes, número de palavras etc.) e palavras-chave específicas. A extração das métricas utilizou validação cruzada, e os indicadores reportados foram: Precisão, Sensibilidade e Kappa. Para treino, foi utilizado um pequeno número de amostras (apenas 61 no total).

Além dos artigos citados, outros autores brasileiros também apresentaram soluções em estimativa de esforço, entre eles Neto [98] e Sitonio [134] [98, 134].

Resumidamente, apresenta-se a seguir um resumo para cada uma dessas duas investigações.

Neto [98] criou um preditor utilizando a experiência do desenvolvedor e a complexidade da User Story. A complexidade foi gerada por meio da avaliação dos próprios programadores; a experiência foi gerada a partir da razão entre a quantidade de projetos em que o desenvolvedor já participou e a quantidade de *User Stories* já realizadas. Porém, o estudo não utilizou o texto da *User Story* no modelo preditivo. Ele avaliou três algoritmos diferentes: Regressão Linear, Árvore de decisão e Floresta Aleatória. Como métrica, foram utilizados o MSE e o RMSE. Aparentemente, a árvore de regressão teve o melhor resultado.

Sitonio [134] criou um preditor de estimativa de esforço com dados de uma empresa real. Ele utilizou AutoML para a seleção do algoritmo e dos hiperparâmetros do modelo; foi especificamente utilizada a biblioteca de código aberto TPOT. A métrica utilizada foi o RMSE. O algoritmo com melhor desempenho foi o *LinearSVR*. Técnicas de PLN foram aplicadas aos atributos preditores textuais (título e descrição).

C.3.2 Extração dos dados

Um resumo com o resultado da extração dos dados é apresentado na Tabela C.6. Nessa tabela, a coluna “Alvo” é o atributo alvo (também chamado de variável dependente) do modelo preditivo proposto nos artigos. A coluna “Preditor” consiste nos atributos preditores (ou independentes) utilizados no modelo. “Métricas” são as métricas utilizadas para avaliação dos modelos e “Técnicas” são algumas das técnicas utilizadas.

O *Story Point* foi o atributo alvo (ou variável dependente) mais utilizado nos estudos selecionados, conforme a Figura C.3 que quantifica o atributo alvo utilizado nos artigos.

A *User Story* foi o atributo preditor (ou variável independente) mais utilizado entre os artigos selecionados. Conforme a Figura C.4 que apresenta um contador com o atributo preditor utilizado nos artigos selecionados.

Figura C.3: Atributo alvo (saída) utilizado nos artigos selecionados.

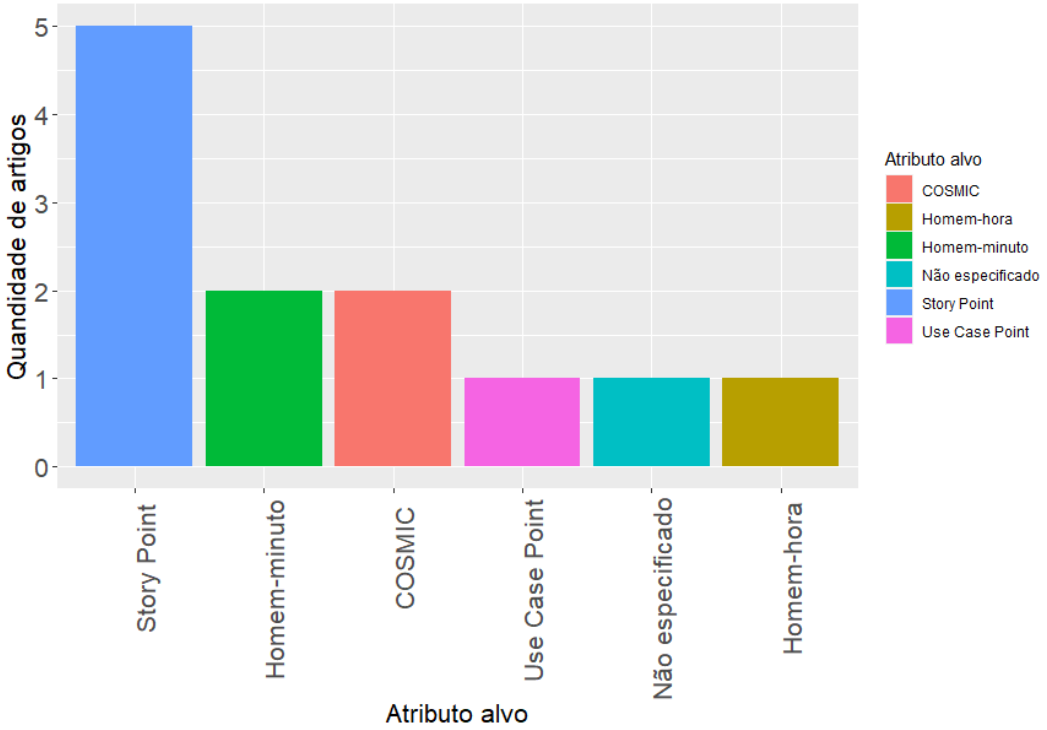
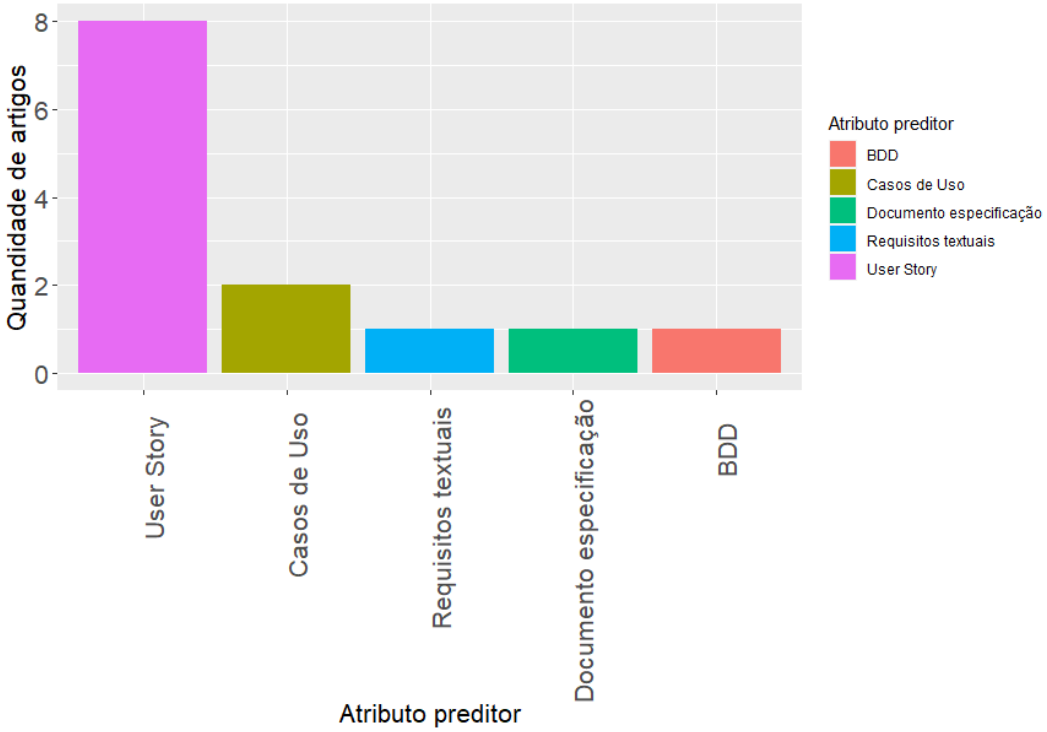
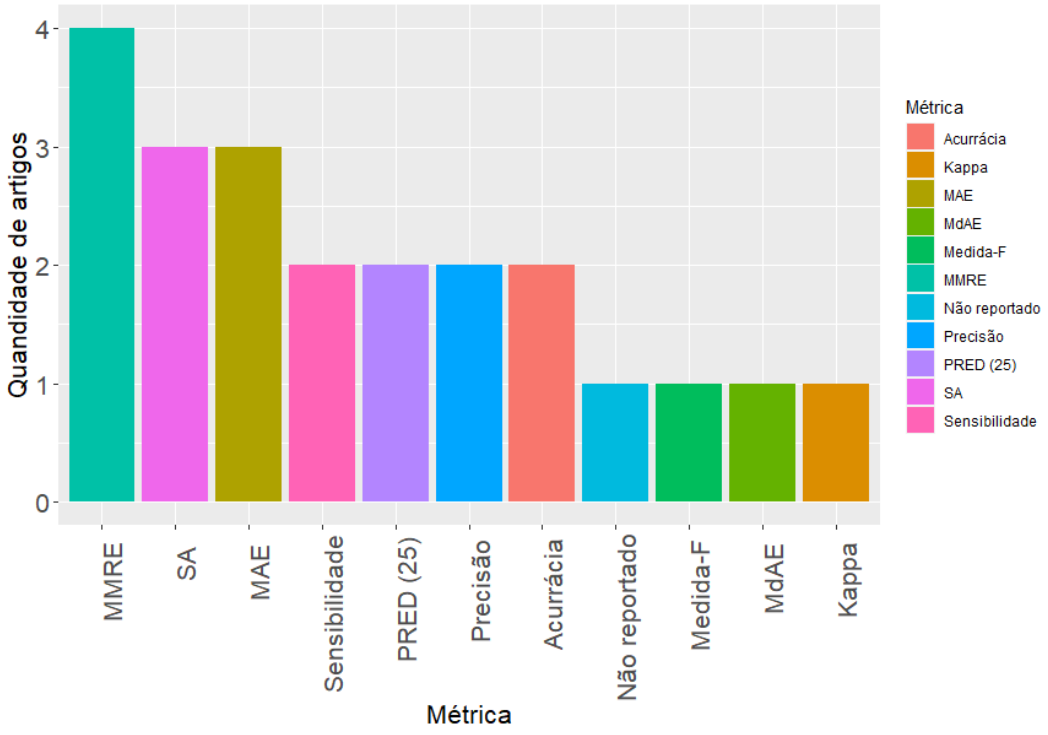


Figura C.4: Atributo preditor utilizado nos artigos selecionados.



Em relação as métricas, MMRE, MAE e SA foram as mais utilizados entre os artigos selecionados. Conforme a Figura C.5 que resume as métricas utilizadas nos artigos selecionados.

Figura C.5: Métrica utilizada nos artigos selecionados.



As técnicas de PLN e AM utilizadas foram agrupadas na Tabela C.6.

C.4 Revisões da literatura relacionadas

Essa seção apresenta outras revisões sistemáticas já realizadas por outros pesquisadores. Elas são apresentadas na Tabela C.7 tendo sido encontradas com a técnica de *snowballing* [133] aplicadas nos estudos primários encontrados. A coluna Cit. é a quantidade de citações que a revisão teve consultadas a partir do Google Acadêmico.

#R1 [131]: Eles realizaram uma revisão em 1.178 artigos. Utilizaram a técnica de PLN *Latent Dirichlet allocation* para extrair padrões desse grande conjunto de artigos publicados entre 1996 e 2016. O objetivo foi responder quais os métodos de pesquisa e quais áreas demandam mais atenção no tema *software effort estima-*

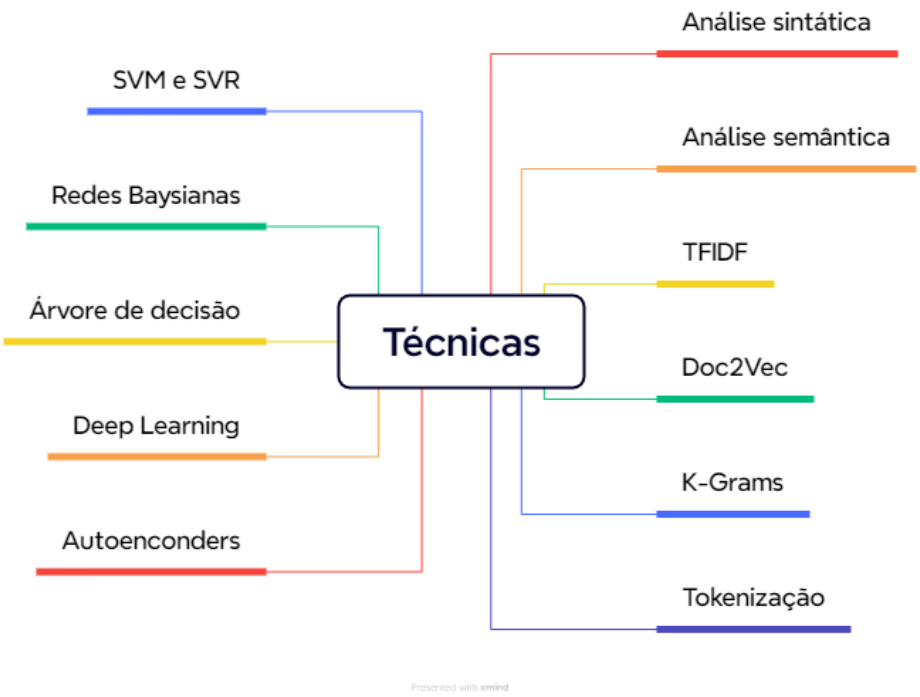
Tabela C.6: Resultado da extração de dados.

# Artigo	**Alvo**	**Técnicas**	**Preditor**	**Métricas**
#1 [31]	Homem-hora	Aprendizado Profundo	*User Story*, outros	MMRE
#2 [106]	Use Case Point	Análise semântica	Casos de Uso	Não reportado
#3 [41]	Homem-minuto	Análise sintática	*User Story*	PRED (25), SA
#4 [23]	*Story Point*	Aprendizado Profundo, LSTM, RNN	*User Story*	MAE, MdAE e SA
#5 [119]	*Story Point*	Análise sintática, probabilidade condicional, Naive Bayes	Narrativas BDD	Acurácia
#6 [130]	*Story Point*	Análise Sintática, TF-IDF	*User Story*	Acurácia, MAE, SA
#7 [135]	*Story Point*	Bag-of-Words, Autoencoders, SVM	*User Story*	Precisão, Sensibilidade e Medida-F
#8 [62]	Homem-minuto	TF-IDF, SVM	*User Story*	MMRE
#9 [111]	*Story Point*	TF-IDF, SVM	*User Story*	MMRE
#10 [100]	COSMIC	Aprendizado Profundo, word embeddings	Casos de uso	MAE
#11 [173]	Não especificado	Análise semântica, Entity, Extração de atributos	Documento de especificação de requisitos	MMRE PRED (25)
#12 [59]	COSMIC	Análise sintática	Requisitos Textuais	Precisão, Sensibilidade, Kappa

Tabela C.7: Outras revisões encontradas via *snowballing*.

Id	Título	Ano	Cit*
#R1	Research patterns and trends in software effort estimation [131]	2017	116
#R2	Systematic literature review of ensemble effort estimation [60]	2016	155
#R3	Seizing Requirements Engineering Issues through Supervised Learning Techniques [52]	2020	17
#R4	Natural Language Processing (NLP) for requirements engineering: A systematic mapping study [174]	2021	231
#R5	Intelligent software engineering in the context of agile software development: A systematic literature review [108]	2020	74
#R6	A review article on software effort estimation in agile methodology [141]	2021	14
#R7	A review of effort estimation studies in agile, iterative and incremental software development [99]	2013	46
#R8	Effort Estimation in Agile Software Development: An Updated Review [32]	2018	37
#R9	Empirical studies of agile software development: A systematic review [39]	2008	3906
#R10	Systematic literature review of machine learning based software development effort estimation models [168]	2016	597

Figura C.6: Técnicas utilizadas nos artigos selecionados.

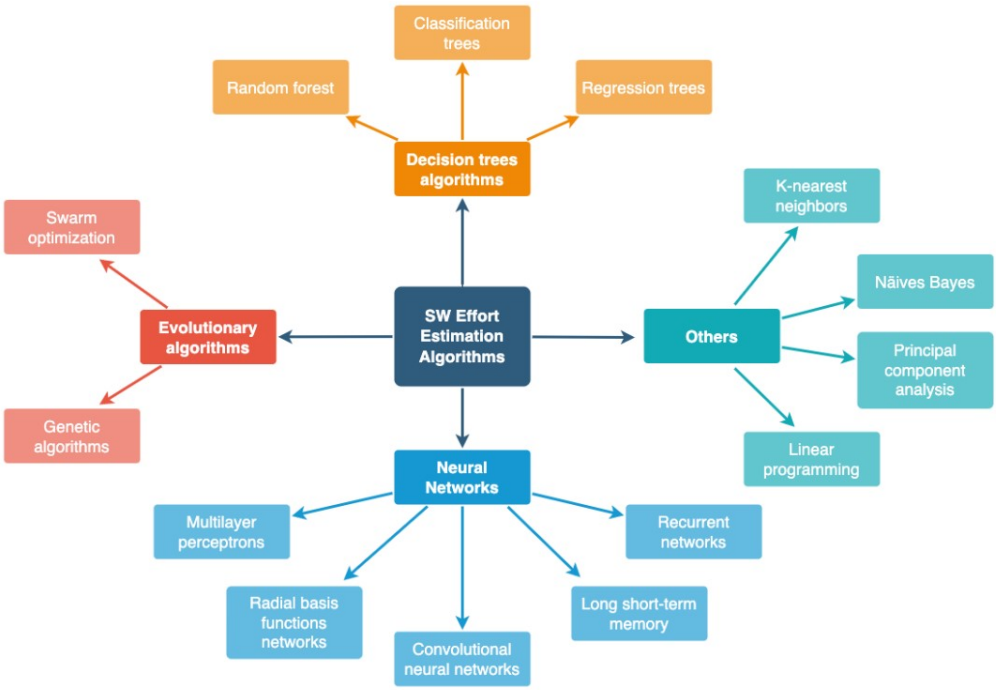


tion. As principais áreas que demandam atenção foram estimativas, validação dos modelos de estimativa e aprendizagem de máquina. As principais técnicas foram *case-based reasoning, support vector regression, neural networks, genetic algorithms, fuzzy logic, Windowing approach, effort estimation model, Ensemble models e use case-based estimation*.

#R2 [60] realizou uma revisão dos métodos "*ensemble*", ou seja, utilizados em conjunto, na estimativa de esforço e concluiu que geralmente os métodos em conjunto performam melhor que métodos únicos.

Além das revisões mencionadas, cabe ressaltar as técnicas levantadas por Cetina (2023) [161] apresentadas na Figura C.7. Ela agrupou as técnicas em 4 grupos, fazendo uma separação coerente das técnicas.

Figura C.7: Técnicas utilizadas em outro estudo.



C.5 Ameaças à validade

Confiabilidade da pesquisa: Utiliza-se uma quantidade pequena de bases de busca de artigos (somente 5). É possível que não tenham sido encontrados todos os artigos relevantes. Para mitigar essa ameaça, foram selecionadas as bases comumente utilizadas em estudos desse tipo. Outra forma de mitigar essa ameaça é um refinamento nas palavras-chave, incluindo sinônimos, o que aumentaria a quantidade de artigos avaliados e possivelmente aceitos.

Confiabilidade da seleção do estudo: Para garantir que a seleção do estudo fosse a mais precisa possível, o mais livre de viés de pesquisador e de erros humanos, seguiu-se um rigoroso processo de seleção de estudos, orientado pelos critérios de inclusão e exclusão definidos com o apoio de uma ferramenta de revisão sistemática, registrando o protocolo e todos os passos da revisão antes da realização da revisão.

Confiabilidade da extração e classificação dos dados: O esquema de extração dos dados necessários envolveu interpretações subjetivas e decisões dos pesquisadores. A falta de alguns detalhes no texto muitas vezes dificultava a extração de

dados. Além da falta de padrão nas tecnologias nos estudos relatados. Além disso, apenas analisar quantitativamente qual foi a métrica mais utilizada não estabelece que essa é a melhor métrica. O mesmo raciocínio se aplica aos atributos preditores, ao atributo alvo ou as técnicas.

C.6 Considerações finais

O objetivo deste apêndice foi identificar as técnicas de PLN utilizadas, os artefatos preditores e os métodos/técnicas, bem como as métricas mais utilizadas na estimativa de esforço. Foi identificado que as técnicas TF-IDF com SVM são mais populares nesse contexto; o artefato de entrada mais utilizado é a User Story; já as métricas de avaliação mais utilizadas foram o MMRE, o MAE e o SA. No geral, embora a área do tema tenha recebido atenção continuada — e, mais recentemente, ascendente — ainda há necessidade de estudos mais empíricos, com novas oportunidades para contribuições científicas.

Apêndice D

NeoDataset: um conjunto de dados com *User Stories* e Story Points

Artigo publicado na conferência “XIII Simpósio Brasileiro de Tecnologia da Informação – SBTI” com o título “NeoDataset - Um conjunto de dados com *User Stories* e Story Points” e, posteriormente, selecionado para ser publicado na “RMP v13 n2 (2024) (Revista dos Mestrados Profissionais)” [95].

RMP
Revista dos Mestrados Profissionais
UFPE / CCSA – MGP
https://periodicos.ufpe.br/revistas/rmp
ISSN – 2317 - 0115
V. 13 - n. 2 (2024)
Recife – PE

NeoDataset: um conjunto de dados com *user stories e story points*

Giseldo da Silva Neo – Instituto Federal de Alagoas
giseldo.neo@ifal.edu.br
https://orcid.org/0000-0001-5574-9260
Alana Viana Borges da Silva Neo - Instituto Federal do Mato Grosso do Sul
alana.neo@ifms.edu.br
https://orcid.org/0009-0000-1910-1598
Kleber Jose Araújo Galvão Filho – Universidade Federal de Alagoas
kleber.filho@ifal.edu.br
https://orcid.org/0009-0001-1591-0272
José Antão Beltrão Moura– Universidade Federal de Campina Grande
antao@computacao.ufcg.edu.br
https://orcid.org/0000-0002-6393-5722
Olival de Gusmão Freitas Junior – Universidade Federal de Alagoas
olival@ic.ufal.br
https://orcid.org/0000-0003-4418-8386

Resumo – As equipes geralmente utilizam ferramentas de gerenciamento para acompanhar as *user stories*, controlar o seu código-fonte, registrar suas estimativas de esforço e os responsáveis. Essas ferramentas registram dados que podem ser utilizados em diversas pesquisas. Por outro lado, é desafiador encontrar dados para pesquisas, pois as empresas privadas são relutantes em compartilhá-los. O objetivo deste artigo é apresentar um conjunto de dados contendo dados brutos de 33 Projetos de *Software* Ágil de código aberto, minerados do GitLab, totalizando 122.627 *story points* e 20.474 *user stories*. Disponibilizamos esses dados publicamente para facilitar o seu uso pela comunidade científica. Acreditamos que esse conjunto pode ser utilizado em várias linhas de pesquisa de engenharia de *software*, incluindo classificação e vetorização de texto, aprendizagem de máquina, estimativa de esforço e priorização de tarefas.

Palavras-chave: *user story , story points*, conjunto de dados, ágil, linguagem natural.

NeoDataset - A dataset with user stories and story points

Abstract – Teams often use management tools to monitor outstanding User Stories, control their source code, record their effort estimates and those responsible for opening and closing tickets. These tools contain data that can be used in various software engineering research. It is necessary to find data for research as private companies are reluctant to share their data. This paper aims to present a dataset containing raw data from 33 open-source Agile Software Projects, mined from GitLab, totaling 122.627 Story Points and 20.474 User Stories. We make this data publicly available in CSV and JSON formats to facilitate its use by the interested scientific community. We believe this dataset can be used in multiple lines of software engineering research, including effort estimation.

Keywords: user story, story points, dataset, agile, natural language.

Data da Submissão: 24/08/2024 - **Data de aceitação:** 01/12/2024

DOI: https://doi.org.10.51359/2317-0115.2024.265431

RMP
Revista dos Mestrados Profissionais
UFPE / CCSA – MGP
https://periodicos.ufpe.br/revistas/rmp
ISSN – 2317 - 0115
V. 13 - n. 2 (2024)
Recife – PE

1. Introdução

Uma *user story* é uma técnica no desenvolvimento ágil de software, que visa capturar as necessidades dos usuários de maneira clara e acessível, facilitando a comunicação entre os membros da equipe de desenvolvimento e os stakeholders (COHN, 2005). Elas são escritas em uma forma narrativa que geralmente segue um formato simples: "Como [tipo de usuário], eu quero [funcionalidade], para que [benefício]". Este formato ajuda a garantir que o foco permaneça no valor que o usuário final obterá da funcionalidade solicitada, ao invés de nos detalhes técnicos de implementação. Ao representar requisitos de forma concisa e orientada ao usuário, as *user stories* permitem uma abordagem incremental e iterativa, adaptada às mudanças frequentes e melhorias contínuas.

Por outro lado, os *story points* são uma métrica usada para estimar o esforço necessário para implementar uma *user story*. Ao contrário das estimativas de tempo tradicionais, os *story points* se concentram na complexidade relativa de um conjunto de tarefas, incluindo o volume de trabalho, incertezas e riscos associados (COHN, 2005). Essa estimativa evita problemas relacionados à precisão inerente às estimativas de tempo e reconhece que diferentes equipes terão diferentes capacidades e velocidades de trabalho. Geralmente, a pontuação é atribuída usando uma escala numérica baseada na sequência de Fibonacci (1, 2, 3, 5, 8, e assim por diante), o que ajuda a evitar debates detalhados sobre pequenas diferenças de esforço e promove um consenso mais rápido dentro da equipe.

Uma *user story* bem elaborada fornece a base necessária para que a equipe de desenvolvimento possa discutir e compreender a tarefa, enquanto os *story points* registram a estimativa e são utilizadas para o planejamento do sprint. Esse processo, conhecido como planejamento de release ou de sprint, permite que a equipe visualize o trabalho a ser realizado em pequenas iterações, mantendo o alinhamento com as prioridades dos stakeholders e ajustando-se rapidamente às mudanças no escopo ou nos requisitos. Além disso, ao utilizar *story points*, equipes podem medir e melhorar sua velocidade ao longo do tempo, refinando suas estimativas e adaptando-se à realidade do projeto, resultando em entregas mais previsíveis e de maior qualidade (SCHWABER; SUTHERLAND, 2020).

A relevância de dados consolidados em pesquisas é amplamente reconhecida; no entanto, empresas privadas frequentemente demonstram relutância em divulgá-los. Este comportamento pode ser associado a vários motivos, incluindo preocupações com a privacidade, questões competitivas e o receio de que os dados possam ser mal interpretados ou utilizados fora do contexto (SMITH, 2017). Além disso, mesmo quando as empresas decidem compartilhar seus dados, nem sempre esses materiais são disponibilizados em um formato que facilite o acesso e a análise pelos pesquisadores (JOHNSON, 2019). Em diversos casos, os dados são desestruturados ou armazenados em sistemas proprietários que exigem ferramentas específicas e conhecimento especializado para a sua extração e manipulação (KIM; PARK, 2020).

O processo de consolidação e preparação dos dados, que é fundamental para transformar dados brutos em informações úteis e interpretáveis, ainda requer um esforço considerável. Esse desafio é amplificado pelo fato de que os dados frequentemente são armazenados em bancos de dados relacionais complexos. A extração de dados desses sistemas pode ser uma tarefa trabalhosa e demorada, envolvendo a escrita de consultas específicas em linguagens como SQL (*Structured Query Language*), a limpeza de dados para remover inconsistências e a integração de dados provenientes de múltiplas fontes. Esses passos são essenciais para garantir que os pesquisadores possam conduzir análises robustas e tirar conclusões precisas.

O acesso facilitado a dados de qualidade constitui uma componente frequente nas discussões sobre avanços na pesquisa científica, especialmente em domínios como a ciência de dados, economia e saúde (JOHNSON, 2019). No entanto, a mediação entre as necessidades dos pesquisadores e as condições impostas pelas empresas para o compartilhamento de dados continua a ser uma área que demanda atenção constante e soluções inovadoras para maximizar o aproveitamento do potencial informacional disponível (MAYER-SCHÖNBERGER, 2013; HARDT, 2019).

Para contribuir com pesquisas de engenharia de software e inspirados pela contribuição de outros conjuntos de dados (JUST, 2014; TAWOSI, 2022), disponibilizamos um novo conjunto de dados chamado NeoDataset. Ele é disponibilizado no GitHub para que toda a comunidade interessada possa contribuir, semelhante ao que acontece com outros conjuntos de dados (TOMASSI, 2019).

O objetivo deste artigo é apresentar um conjunto de dados com dados textuais do título e da descrição da *user story* e sua medição em *story points* realizada pelo time. Os dados coletados abrangem diversas equipes de desenvolvimento ao longo de vários projetos que podem revelar padrões e insights sobre práticas de estimativa e planejamento e esforço. Este novo conjunto de dados constitui uma fonte de informações para problemas de classificação de texto e para aprimorar a precisão das estimativas em projetos ágeis.

As seções seguintes apresentam a fundamentação teórica, a descrição do conjunto de dados, como ele foi extraído, suas características e estrutura, a sua originalidade e relevância e as considerações finais.

2. Referência Conceitual

Esta seção apresenta a *user story* e o *GitLab*, conceitos necessários para o entendimento de outras seções do artigo.

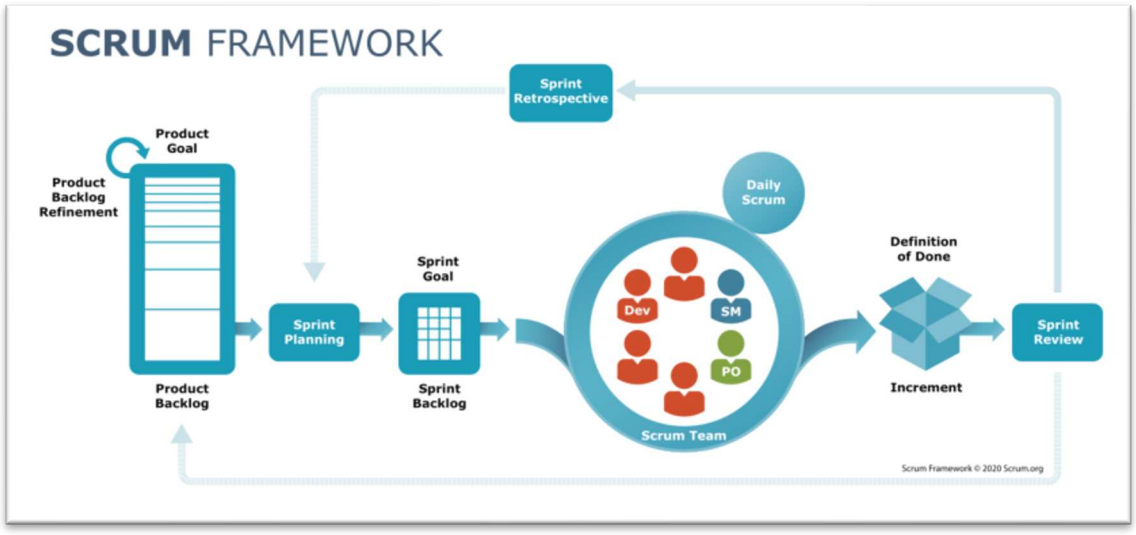
2.1 User story

Em 1993 o engenheiro de *software* Jeff Sutherland criou um processo de desenvolvimento chamado SCRUM (SUTHERLAND, 2014). Jeff trabalhava na empresa Easel Corporation e aplicou o SCRUM no projeto mais crítico dessa organização, obtendo bons resultados (SABBAGH, 2014). Em meados de 2001, ele, Ken Schwaber, Kent Beck e outros 16 pesquisadores discutiram as dificuldades no processo de desenvolvimento tradicional. No final desse encontro eles elaboram o manifesto ágil (BECK, 2001). Esse manifesto deu início ao movimento ágil, do qual o SCRUM faz parte.

O SCRUM é um método ágil baseado em ciclos de tempo fixo chamados *sprints*. Em uma *sprint* o time trabalha para atingir objetivos bem definidos. Esses objetivos são registrados em uma lista de coisas a fazer que é constantemente atualizada e re-priorizada, chamada *product backlog* (SUTHERLAND, 2014).

A característica do SCRUM é o foco em entregas rápidas com alto valor agregado em curtos períodos, lidando com as mudanças o mais rápido possível (DYBA, 2008). Esta abordagem tem mostrado resultados e tem sido utilizada em gerenciamento de projetos (PMI, 2017), tanto na indústria (TRIMBLE, 2016; RIGBY, 2018) quanto no governo (MERGEL, 2016) - Vide figura 1.

Figura 1 – Scrum Framework.



Fonte: Scrum Aliance (2024).

As *user stories* são peças centrais no registro de requisitos em times que usam o SCRUM. Essa narrativa deve ser concisa e descrever a funcionalidade desejada por um usuário em um sistema (COHN, 2009). Elas podem ser descritas em diferentes níveis de granularidade, desde funcionalidades abrangentes, até detalhes específicos da interface. Também podem ser estruturadas de diversas maneiras, porém, uma estrutura formal comumente utilizada é a proposta por (COHN, 2009) em três elementos principais:

- Quem: O tipo de usuário que necessita da funcionalidade;
- O que: A funcionalidade específica que o usuário precisa;
- Porque: A razão pela qual o usuário precisa da funcionalidade e os benefícios que ela trará.

A empresa *Mountain Goat Software* fornece 200 exemplos de *user stories* que podem ser utilizados para treinamentos ou para estudos. Essas *user stories* foram escritas durante a construção do *site* da Scrum Alliance entre 2003 e 2004. Elas estão disponíveis para *download* em formato PDF (MOUNTAIN, 2004). O quadro 1 apresenta alguns exemplos desse conjunto de dados.

Outros conjuntos de dados com *user stories* podem ser encontrados no *site* Mendeley Data. Esse *site* é um repositório que armazena vários conjuntos de dados

disponíveis para pesquisas. Um dos conjuntos de dados existentes nesse repositório consolida *user stories* de 22 projetos (DALPIAZ, 2018). O quadro 2 apresenta exemplos de *user stories* deste conjunto de dados.

Note que a descrição do texto da *user story* nos quadro 1 e quadro 2 segue a estrutura: "quem", geralmente utilizando a palavra "como"; também a estrutura "o que", geralmente utilizando as palavras: "quero", "posso", "desejo"; Por fim, a estrutura "Porque" com a palavra "para".

Quadro 1 - Exemplos de *user stories* do site Mountain Goat Software.

Descrição
Como membro do *site*, quero descrever-me na minha própria página de uma forma semiestruturada para que os outros possam aprender sobre mim. Ou seja, posso preencher campos predefinidos, mas também ter espaço para um ou dois campos de texto livre. (Seria bom deixar esse texto livre ser *Markdown* ou similar para permitir alguma formatação simples)
Como membro do site, posso preencher uma inscrição para me tornar um *Certified Scrum Practitioner* para que eu possa ganhar essa designação. *Certified Scrum Practitioner* foi o nome inicial do que ficou conhecido como *Certified Scrum Professional*
Como *Practitioner*, quero que minha página de perfil inclua detalhes adicionais sobre mim (ou seja, algumas das respostas à minha inscrição no *Practitioner*) para que eu possa mostrar minha experiência

Quadro 2 - Conjunto de dados com várias User Stories de Dalpiaz, disponível no Mendeley Data.

Descrição
Como designer de interface do usuário, desejo reprojetar a página Recursos para que ela corresponda aos novos estilos de design do *Broker*.
Como um designer de interface do usuário, quero relatar às agências sobre testes de usuários, para que eles estejam cientes de suas contribuições para tornar o *Broker* uma UX melhor.
Como designer de interface do usuário, quero passar para a rodada 2 de edições de página de destino DABS ou FABS, para que eu possa obter aprovações da liderança.

2.2 GitLab

O *GitLab* (https://gitlab.com/) é uma plataforma de desenvolvimento de *software* que combina diferentes fases do ciclo de vida de uma aplicação. Inicialmente concebido como um sistema de controle de versão utilizando *Git*, ele evoluiu para um ambiente completo que apoia desde a gestão de repositórios até a entrega contínua (KONONOV, 2018; MURPHY et al., 2020).

Entre as funcionalidades disponibilizadas pelo *GitLab*, destacam-se a gestão de projetos e repositórios, a integração contínua, a entrega contínua e a integração com outras ferramentas e serviços. A plataforma permite a colaboração entre equipes de desenvolvimento por meio de um sistema de *merge requests*, revisão de código e rastreamento de problemas (*issues*), tudo centralizado em uma interface única.

O *GitLab* pode ser utilizado tanto na nuvem, por meio de seu serviço hospedado, quanto em instalações *on-premises*, permitindo às organizações escolherem o modelo que melhor se adequa às suas necessidades de segurança e controle. Adicionalmente, o *GitLab* incorpora funcionalidades como monitoramento de desempenho, segurança de aplicações

e automação de *deploys*, oferecendo um ambiente coeso para todo o ciclo de vida do desenvolvimento. Essas características fazem do *GitLab* uma ferramenta abrangente para equipes de desenvolvimento que buscam uma integração eficiente de processos em um único arsenal de software.

A maioria das equipes que utilizam *user stories* também utiliza ferramentas de *software* para gerenciar o projeto e principalmente para manter um registro de suas *user stories* (JADHAV, 2023). Ao analisar os dados registrados por essas ferramentas podemos extrair informações para diversas pesquisas de engenharia de *software*, incluindo pesquisas sobre como melhorá-las (JIMÉNEZ et al., 2023).

O *GitLab* é uma das ferramentas de gerenciamento utilizadas por equipes ágeis para registrar *user stories* (CHOUDHURY et al., 2020). Ele permite que os engenheiros de *software* automatizem muitas ações durante o ciclo de desenvolvimento, incluindo registrar e alterar *user stories* (DIMITRIJEVI et al., 2015). No *GitLab*, a *user story* é registrada como um *issue*, e para cada *user story* são armazenadas diversas informações, como o título, a descrição e sua estimativa em *story points*. Outra ferramenta utilizada com funcionalidades semelhantes é o Jira® da empresa Atlassian.

Figura 2 - Quadrante mágico Gartner - *GitLab*

Fonte: Gatner (2024).

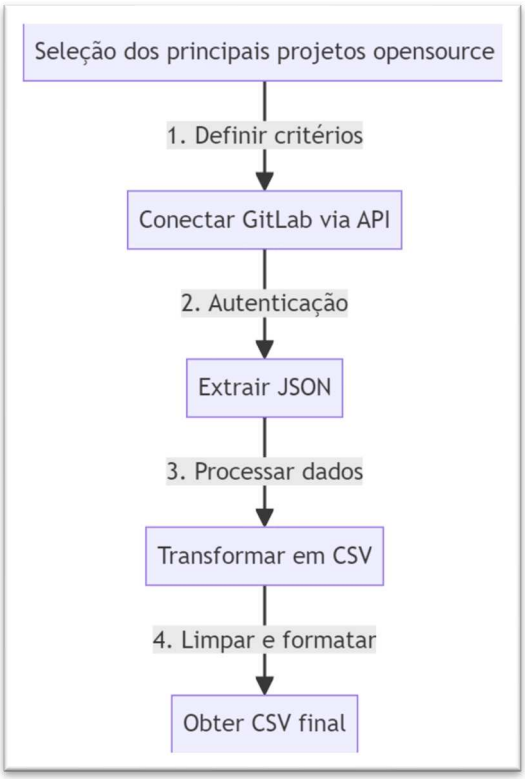
Esses dados vem sendo utilizados em diversas pesquisas relacionadas a problema em engenharia de software, tais como, atribuição de *user stories* (MANI et al., 2019), melhoria da descrição de *user stories* (CHAPARRO et al., 2017), planejamento da *sprint* (CHOETKIERTIKUL et al., 2018), análise de sentimento de desenvolvedores que escrevem as *user stories* (ORTU et al., 2015; ORTU et al., 2016; VALDEZ et al., 2020) estimativa de esforço de *user stories* (PORRU et al., 2016; SOARES et al., 2018, DRAGICEVIC et al., 2017; CHOETKIERTIKUL et al., 2019; TAWOSI et al., 2022) e

priorização de *user stories* (GAVIDIA-CALDERON et al., 2021; HUANG et al., 2021 e UMER et al., 2020). Além disso, o *GitLab* em 2023 está localizado no canto superior direito do quadrante Gartner como uma ferramenta líder em *DevOps* (figura 2). Os líderes do Quadrante Mágico das Plataformas de *DevOps* de 2023 influenciam a direção do espaço de *DevOps* com sua liderança de pensamento e serviços. Eles representam plataformas funcionalmente ricas com vários recursos e suportam o *software* durante todo o ciclo de vida de desenvolvimento.

3. Metodologia

O procedimento metodológico é composto por quatro etapas principais. Inicialmente, foram definidos os critérios de seleção dos projetos, eles foram filtrados pelo número de estrelas do repositório. Na segunda etapa, realizou-se a autenticação, e os dados foram extraídos no formato original (JSON). A terceira etapa envolveu o processamento desses dados. Finalmente, na quarta etapa, os dados foram agrupados resultando na obtenção do arquivo final no formato CSV, pronto para análise ou integração em outras plataformas, veja na figura 3. Cabe ressaltar que não foi realizado nenhum processamento de dados brutos antes da disponibilização do CSV.

Figura 3 – Procedimentos metodológicos.



Como resultado, foi produzido/consolidado um conjunto de dados que abrange informações de 33 projetos de desenvolvimento de *software*, com 20.474 *user stories* retiradas de repositórios do *GitLab*, totalizando 122.627 *story points*. Apenas *user stories*

com o atributo *state* igual a *closed* e que possuem o atributo *weight* preenchido foram coletadas. O campo *weight* é utilizado no *GitLab* para registrar o esforço em *story points*.

O conjunto de dados apresentado inclui projetos que não foram utilizados por estudos anteriores, de acordo com nossas buscas não sistemáticas em artigos. Já existem estudos prévios que extraíram dados da ferramenta de gestão *Jira* em diversas pesquisas de engenharia de *software* (TAWOSI et al., 2022; CHOETKIERTIKUL et al., 2019; PORRU et al., 2016 e SCOTT et al., 2018), contudo, projetos extraídos do *GitLab* são mais raros.

A reprodução de outros estudos, tal como o de Venkatraman *et al.* (2024) pode se beneficiar deste conjunto de dados, pois seus resultados podem ser reproduzidos para fortalecer a evidência sugerida, por exemplo de que determinado algoritmo é mais adequado para a finalidade de estimativa de esforço.

4. Resultados

4.1 Extração

Este conjunto de dados foi extraído durante o mês de janeiro de 2023 e abril de 2023. O processo de mineração teve como alvo os principais projetos de código aberto do *GitLab*. Os projetos selecionados empregam metodologias ágeis de desenvolvimento de *software* e tiveram registrado o tamanho em *story points* de suas tarefas pela equipe de desenvolvimento do projeto para minerar informações do *GitLab* foi criado uma ferramenta de extração implementada em *Python* que se conecta ao *GitLab* via API, o código dessa aplicação está disponível no *GitHub* em https://github.com/giseldo/neo-gitlab-extractor.

A figura 4 apresenta um exemplo no conjunto de dados. Ela contém as informações do "Título", "Descrição" e "SP" de três *user stories*. Na coluna "Descrição", são apresentados detalhes sobre cada problema ou tarefa. A primeira linha é a *user story* "*Posts repeat in Groups feed*" ela descreve um problema em que posts se repetem em um feed de grupos, o que parece ser uma questão em uma instância de teste. Detalhes técnicos sobre o comportamento e possíveis soluções são discutidos, como criar um novo grupo e postar uma sequência de *posts* para replicar o problema.

Ainda na figura 4, a segunda linha é a *user story* "*Expose bot accuracy scores on front end for admins*" que fala sobre a necessidade de ajudar os administradores a tomar decisões informadas ao exibir as pontuações de precisão de um *bot* na interface de administrador, juntamente com uma descrição de como essas informações devem ser apresentadas, incluindo números ao lado do avatar do usuário. A terceira linha é a *user story* "*Refactor experiments to use events instead of contexts*" ela discute a necessidade de reformular como os experimentos são realizados, sugerindo que, em vez de basear experimentos em contextos como o carregamento de uma página, eles sejam baseados em eventos específicos que os usuários desencadeiam, com propostas para melhorias no processo. Por fim, na coluna "SP", são atribuídos números de prioridade (5, 3, 8) a cada uma das tarefas descritas, indicando o esforço de cada uma delas.

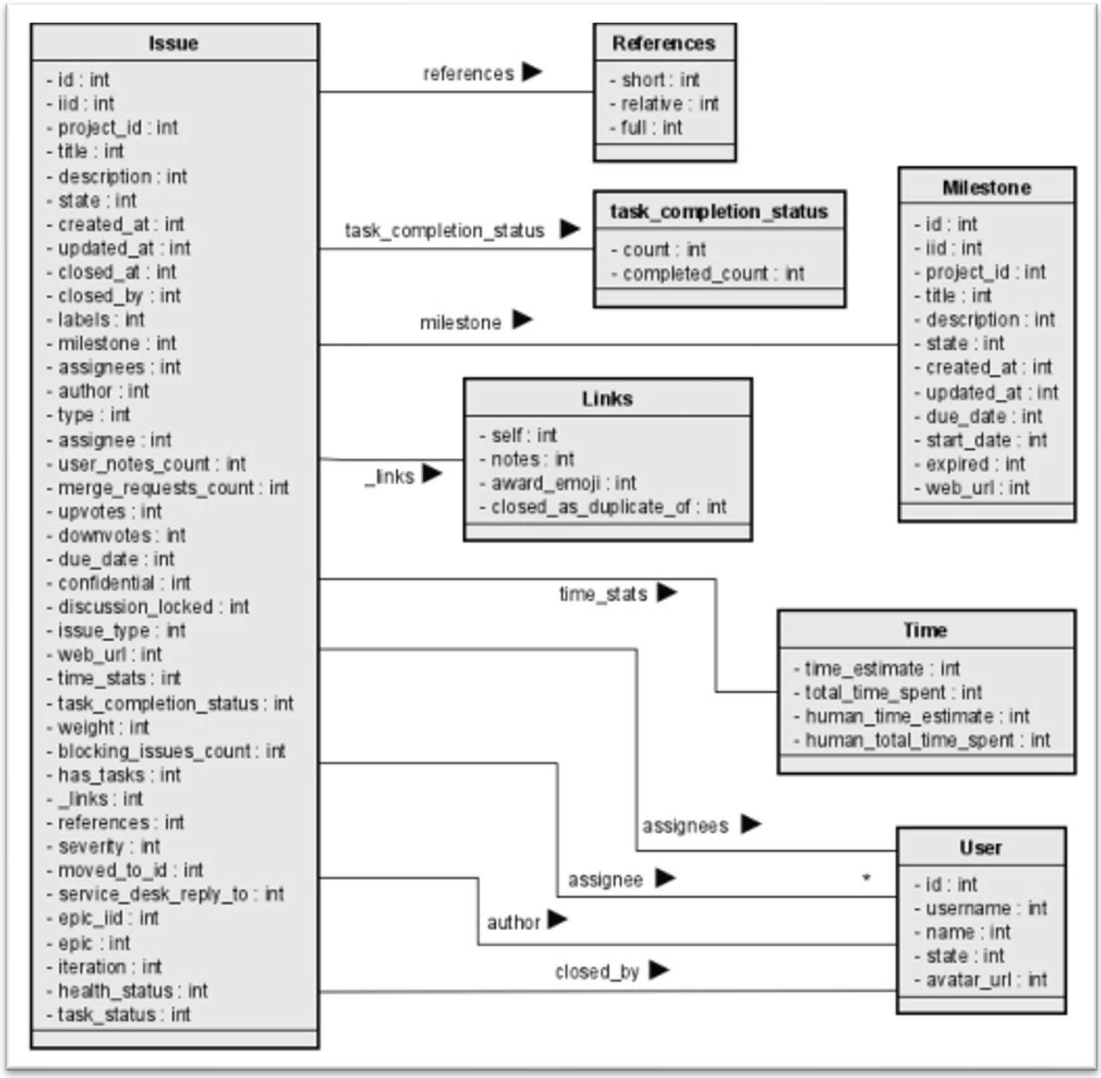
Figura 4 – Exemplo do conjunto de dados.

Título	Descrição	SP
Posts repeat in Groups feed	We have a few reports of posts repeating in groups. [This Gitlab issue](url...) makes reference to the issue in a test Group. Additionally, a user alerted me that [this Group](url...) exhibits the issue, though notably the posts repeat only after scrolling VERY deep into the timeline (~100+ posts). The pinned comment re-appears in the feed, after which the feed appears to loop. Note timestamps on the posts in this sequence: ![image](url...) ![image](url...) ![image](url...) ### Replication steps Effect can be seen here: (url...) To replicate a base-case: 1. Make a new group 2. Make a post titled 1, and a post titled 2 3. Refresh	5
Expose bot accuracy scores on front end for admins	In order to (a) help with making admin decisions, and (b) QA check the scores so that admins can see what Tasman is spitting out and provide feedback on the effectiveness of the current scoring, we'd like to make Tasman-derived trust scores visible on the front end of Minds website for users with Admin access. (url...) >Designs are in progress - Given Tasman trust scores for users are available, - and UserA is logged into Minds, - and UserA's account has Admin access, - when any user avatar + name component is displayed, - then a number is displayed alongside the avatar + name that exposes the Tasman trust score for that user. Proposed approach for displaying the correct colour based on a 0-100 value ->(url...) ### Notes * Uniquely an admin feature for the first iteration * In the future this will be visible to all users, including an explainer (via a modal) about the score, why scores are valuable and how to improve scores.	3
Refactor experiments to use events instead of contexts	## The Problem Currently all users are bucketed into experiments on app initialisation and not when they actually see the experiment. This was done so that pageviews could register the experiments as contexts when someone lands on the site. Issues arise when experiment reports rely on only including users who have seen the experiment to be included. For example, an experiment that forwards users to discovery instead of the newsfeed after registration should not just only include users that have just signed up, it should also only apply to users that do not have any referral logic (ie. take newly loggedin/registered users back to previous page). ## Proposed change - [x] Create a new iglu schema `growthbook_event` which includes the experiment_id that the user is in - [x] Remove backend registration and client on init logic - [x] Record a `growthbook_event` via snowplow when an experiment is run - [x] To avoid unnecessary events, cache a reference to the events so that they only get sent every 24 hours. - [x] Now that we support psuedo_id, attach both the anonymous_id and user_id to growthbook events	8

A figura 5 apresenta os dados *JSON* originais que foram coletados com a *API*, porém somente o título, descrição, id do projeto, id da *user story* e data da criação foram consolidados na versão final do conjunto de dados. O diagrama de entidade-relacionamento ilustra a estrutura de dados associada ao gerenciamento da *user story*.

Ainda na figura 5, a entidade central, denominada "*Issue*", engloba atributos essenciais como identificadores, título, descrição, estado, datas de criação e atualização, além de informações sobre o responsável pela tarefa e suas referências. Conectadas a essa entidade principal, estão outras entidades, como "*Task Completion Status*", que monitora o progresso da tarefa, registrando o total de subtarefas e o número concluído; e "*Milestone*", que marca etapas significativas do projeto, oferecendo informações como datas de início e término, além de descrições. Outras entidades incluem "*Links*", que armazena links associados e outras informações complementares, "Time", que acompanha o tempo estimado e o efetivamente gasto na tarefa, e "*User*", que representa os usuários do sistema com seus respectivos identificadores e informações de perfil. As relações entre essas entidades demonstram a interconexão dos dados de uma "*Issue*" com diversos componentes do sistema, refletindo o modelo utilizado no *GitLab*.

Figura 5 – *JSON* Original coletado via *API*, antes da conversão para *CSV*.



4.2 Armazenamento

O conjunto de dados em sua versão final está consolidado em um único arquivo no formato *CSV*. O formato *CSV* é amplamente utilizado para armazenar e trocar dados tabulares de maneira eficiente e simples. Ele é um dos formatos mais comuns e versáteis para armazenar dados tabulares. Devido à sua simplicidade, é adotado em diversas áreas acadêmicas, comerciais e tecnológicas. A principal característica que define um arquivo *CSV* é que ele representa dados em forma de texto, onde cada linha corresponde a um registro ou linha da tabela, e os valores dentro de uma linha são separados por vírgulas.

4.3 Características

Os projetos do conjunto de dados possuem diferentes características, abrangem diferentes linguagens de programação, diferentes domínios de negócio e diferentes localizações geográficas da equipe. Não foi realizado nenhum processo de limpeza ou transformação nas colunas.

O quadro 3 apresenta uma descrição dos atributos (também chamados de variáveis ou colunas) do conjunto de dados além de sua categorização, se quantitativo ou qualitativo. O *idproject* é um número que identifica aquele projeto. A *issuekey* é um número chave que identifica aquela *user story* , o campo *created* é a data/hora de criação da *user story*, o campo *title*, é o título daquela *user story* em linguagem natural, o campo *description* é a descrição da *user story* escrito em linguagem natural, o campo *story point* é a quantidade de *story points* registrada pela equipe de projeto para aquela *user story*.

Quadro 3 – Descrição dos atributos do conjunto de dados.

Nome	Descrição	Categorização
idproject	Chave de identificação do projeto	Qualitativa (campo chave)
issuekey	Chave de identificação da u*ser story*	Qualitativa (campo chave)
created	Data de criação da *User story*	Qualitativa
title	Título da u*ser story*	Texto em linguagem natural
description	Descrição completa da u*ser story*	Texto em linguagem natural
storypoint	Quantidade de *story point* daquela u*ser story*	Quantitativa Ordinal

A tabela 1 apresenta a estatística descritiva do conjunto de dados, incluindo quantidade, média, desvio padrão e percentis, agrupados pelo identificador do projeto. Os projetos utilizam escalas de *story points* distintas; por exemplo, o projeto com o *idproject* 28419588 tem um valor máximo de *story points* de 300, enquanto o projeto 7764 apresenta um valor máximo de 128. Além disso, em alguns projetos, observa-se *story points* com valor zero, o que pode indicar a ausência de esforço associado.

4.4 Disponibilidade

Existem diversas plataformas disponíveis para a disponibilização de conjuntos de dados, entre as quais destacam-se o *HuggingFace* (https://huggingface.co/) e o *Mendeley Data* (https://data.mendeley.com/). O *NeoDataset* está acessível em ambas as plataformas.

O conjunto de dados disponível no *HuggingFace* pode ser acessado no endereço: https://huggingface.co/datasets/giseldo/neodataset. O *HuggingFace* é uma plataforma reconhecida por oferecer uma vasta gama de recursos para a manipulação e compartilhamento de aplicativos e conjuntos de dados relacionados a aprendizagem de máquina. Grandes modelos de linguagem e conjuntos de dados utilizados são acessíveis a partir desta ferramenta.

Os conjuntos de dados hospedados no *HuggingFace* são facilmente acessíveis por pesquisadores e desenvolvedores ao redor do mundo, aumentando a visibilidade e uso de seus dados. Disponibilizar um conjunto de dados no *HuggingFace* pode ser benéfico para a comunidade científica de aprendizagem de máquina, uma vez que pode promover a transparência, a replicabilidade e a inovação.

Tabela 1 – Estatística descritiva.

	count	mean	std	min	25%	50%	75%	max
idproject								
7764	355.0	2.732394	7.186298	0.0	1.00	2.0	3.0	128.0
250833	13.0	2.692308	2.213015	1.0	1.00	2.0	3.0	9.0
734943	121.0	2.190083	2.608684	1.0	1.00	1.0	3.0	20.0
1304532	1724.0	2.621810	2.487977	0.0	1.00	2.0	3.0	21.0
1714548	154.0	2.597403	1.510397	1.0	1.00	2.0	3.0	8.0
2009901	171.0	2.345029	1.460546	1.0	1.00	2.0	3.0	13.0
2670515	1574.0	1.993011	1.226670	0.0	1.00	2.0	2.0	15.0
3828396	15.0	2.066667	1.162919	1.0	1.00	2.0	2.0	5.0
3836952	103.0	26.902913	38.626293	0.0	2.50	5.0	32.0	100.0
4456656	344.0	1.988372	1.551782	0.0	1.00	1.0	3.0	13.0
5261717	106.0	1.745283	0.744009	1.0	1.00	2.0	2.0	4.0
6206924	42.0	2.166667	0.960606	1.0	1.25	2.0	3.0	4.0
7071551	310.0	1.935484	2.005421	0.0	1.00	2.0	2.0	32.0
7128869	327.0	4.143731	3.562003	0.0	2.00	3.0	5.0	21.0
7603319	237.0	4.670886	5.340257	0.0	1.00	4.0	8.0	40.0
7776928	216.0	2.046296	1.156452	1.0	1.00	2.0	2.0	10.0
10152778	521.0	3.059501	4.173901	0.0	1.00	2.0	3.0	80.0
10171263	982.0	3.810591	3.652708	0.0	2.00	3.0	5.0	32.0
10171270	1845.0	3.086721	3.244228	0.0	1.00	2.0	4.0	40.0
10171280	2796.0	2.579757	2.710954	0.0	1.00	2.0	3.0	50.0
10174980	178.0	2.820225	2.759930	1.0	1.00	2.0	3.0	15.0
12450835	424.0	13.674528	22.051256	4.0	6.00	6.0	12.0	260.0
12584701	171.0	6.467836	7.389409	1.0	2.00	4.0	8.0	48.0
12894267	285.0	4.978947	5.026645	0.0	2.00	4.0	6.0	40.0
14052249	167.0	3.047904	2.895381	1.0	2.00	2.0	3.0	20.0
14976868	113.0	6.380531	7.624348	1.0	2.00	4.0	8.0	42.0
15502567	3284.0	10.563946	14.193085	4.0	6.00	6.0	10.0	268.0
19921167	420.0	4.290476	4.026927	1.0	2.00	3.0	5.0	40.0
21149814	1942.0	2.918641	1.507178	1.0	2.00	3.0	3.0	20.0
23285197	284.0	1.785211	1.455947	0.0	1.00	1.0	2.0	13.0
28419588	144.0	123.611111	48.740947	100.0	100.00	100.0	100.0	300.0
28644964	102.0	139.215686	63.177986	100.0	100.00	100.0	200.0	300.0
28847821	1004.0	2.654382	2.017143	0.0	1.00	2.0	4.0	14.0

Fonte: dados da pesquisa (2024).

Além disso, o conjunto de dados disponível no *Mendeley Data* pode ser acessado em https://huggingface.co/datasets/giseldo/neodataset. O *Mendeley Data* é uma plataforma projetada para a preservação e compartilhamento de dados científicos, promovendo a transparência, a reprodutibilidade e a colaboração interdisciplinar. A plataforma permite que pesquisadores carreguem conjuntos de dados de qualquer

tamanho e formato, acompanhados de metadados detalhados que facilitam a descoberta e reutilização.

Ao disponibilizar dados de forma aberta e acessível, os pesquisadores permitem que outros estudiosos possam validar resultados, realizar meta-análises ou até mesmo identificar novas linhas de investigação. Assim, a publicação de dados no ecossistema *Mendeley* e *HuggingFace* contribui para o avanço da ciência aberta e colaborativa, beneficiando tanto os autores originais quanto a comunidade científica ao redor do mundo.

4.5 4.5 Exemplo em *Python*

Um exemplo de como utilizar baixar o repositório do *HuggingFace* com código em linguagem de programação *Python*, com o suporte da biblioteca *Pandas* (PANDAS, 2023) é apresentado na figura 6. O código faz o *download* do conjunto de dados para o computador onde está sendo executado e exibe suas primeiras linhas. A saída do comando é a tabela 2, nesta figura são exibidas as 5 primeiras linhas do conjunto de dados.

Figura 6 – Primeiras colunas do conjunto de dados.

```
1. import pandas as pd
2. df = pd.read_csv("hf://datasets/giseldo/neodataset/issues.csv")
3. df.head()
```

Tabela 2 – Saída das primeiras colunas do conjunto de dados de um do projeto do conjunto de dados.

	idproject	issuekey	created	title	description	storypoints
0	10152778	19541701	2019-03-28 15:04:16.070	(feat): change 'from' email address to 'no-rep...	Emails should not point to info@minds.com	1.0
1	10152778	19273465	2019-03-20 02:11:19.496	(bug): Link previews disappear when editing a ...	### Summary\r\n\r\nWhen editing a status post ...	3.0
2	10152778	18774779	2019-03-04 11:21:14.993	(bug): subscribed blog post duplicates filling...	NaN	2.0
3	10152778	18438523	2019-02-20 16:14:17.411	(bug): Cant delete a reply to a comment on a b...	### Summary\r\n\r\nOn ones own blog post, whil...	5.0
4	10152778	18177704	2019-02-12 16:55:06.846	Counter in group convo-feeds counts 2 seconds ...	### Summary\r\n\r\nWhen a comment is is left i...	5.0

Fonte: Dados da pesquisa (2024).

4.6 Possíveis aplicações na aprendizagem de máquina

Os dados brutos de *user stories* e *story points*, extraídos de repositórios do *GitLab*, podem alimentar algoritmos de aprendizado de máquina. Esse conjunto de dados permite o desenvolvimento de modelos preditivos para estimar o esforço necessário em tarefas ágeis, além de suportar o aprimoramento de técnicas de processamento de linguagem natural para a classificação e priorização de tarefas de software.

O impacto desse tipo de exploração no futuro do trabalho em engenharia de *software* pode ser significativo, já que automatizar previsões de esforço e priorização pode aumentar a eficiência das equipes, permitindo uma melhor alocação de recursos e adaptação às mudanças em projetos complexos.

Um cenário de uso dos dados disponibilizados pode ser a criação de um modelo de aprendizado supervisionado que, ao ser treinado com o NeoDataset, seja capaz de prever *story points* para novas *user stories* com base em sua descrição textual. Essa ferramenta pode ser integrada diretamente em plataformas de gerenciamento de projetos como *GitLab* ou *Jira*, oferecendo sugestões automáticas para equipes de desenvolvimento

ao planejar sprints, facilitando assim o trabalho colaborativo em grandes equipes distribuídas. Como exemplo o *user story tutor* (NEO et al., 2024) que propôs um modelo preditivo utilizando o NeoDataset disponível no *HuggingFace*.

5. Ameaças a validade

As ameaças à validade deste estudo podem incluir:

- Viés de seleção dos projetos: Os projetos selecionados para a extração dos dados não são representativos de todos os projetos de desenvolvimento de software. Isso pode limitar a generalização dos resultados obtidos;
- Viés de seleção das *user stories*: A coleta de *user stories* que estão fechadas (situação *closed*) e possuem atributo de *story Points* preenchido pode introduzir um viés, pois pode excluir uma parte significativa das *user stories* que ainda estão em andamento ou não possuem estimativas feitas pela equipe;
- Precisão dos *story points*: Os dados de *story points* são baseados em estimativas feitas pelas equipes de desenvolvimento. Essas estimativas podem conter erros ou serem influenciadas por fatores subjetivos, o que pode impactar a precisão das análises feitas com esses dados;
- Dependência da ferramenta *GitLab*: A extração dos dados é baseada na integração com a *API* do *GitLab*. Qualquer alteração na *API* ou na estrutura dos dados pode impactar a coleta e a disponibilidade dos dados futuramente;
- Relevância das informações: Embora os dados coletados possam fornecer insights sobre práticas de estimativa e planejamento de esforço, pode haver outros fatores além das *user stories* e dos *story points* que influenciam essas práticas. Portanto, é importante considerar outras fontes de informação para obter uma visão completa do processo de desenvolvimento ágil de *software*;
- Generalização dos resultados: Os resultados obtidos com este conjunto de dados específico podem não se aplicar a todos os contextos de desenvolvimento ágil de *software*. É importante considerar as características específicas dos projetos e das equipes ao interpretar e generalizar os resultados encontrados.

6. Trabalhos relacionados

Diversos trabalhos na área de engenharia de *software* têm se dedicado à criação e disponibilização de conjuntos de dados que apoiam a pesquisa em metodologias ágeis, principalmente no que tange à análise de *user stories* e estimativas de esforço. Um dos exemplos mais notáveis é o trabalho de Just (2014), que disponibilizou um conjunto de dados focado em métricas de código-fonte e bugs, extraídos de repositórios GitHub. Embora relevante, esse conjunto de dados não explora diretamente as *user stories* ou *story points*, deixando uma lacuna para análises mais focadas nas práticas ágeis de desenvolvimento de *software*.

Tawosi *et al*., (2022) apresentou um estudo que extraiu dados do *Jira*, uma das ferramentas mais utilizadas para gestão de projetos ágeis, oferecendo *insights* sobre o

planejamento de sprints e a estimativa de esforço em ambientes corporativos. No entanto, a maioria dos dados coletados por aqueles autores, se restringe a contextos corporativos e privados, onde o acesso aos dados é limitado Em contrapartida, o NeoDataset contribui significativamente ao disponibilizar um conjunto de dados oriundo de projetos de código aberto no *GitLab*, ampliando a possibilidade de pesquisa em ambientes não corporativos e em projetos onde os dados são mais acessíveis e transparentes.

Outro trabalho relevante é o de Porru *et al*. (2016), que investigou a eficácia das estimativas de *story points* utilizando dados extraídos do *Jira*. Similarmente, Scott *et al*., (2018) também exploraram estimativas de esforço. Por fim, vale mencionar que o uso de dados extraídos do GitLab, como no NeoDataset, é menos explorado na literatura em comparação com dados provenientes de ferramentas como o *Jira*. Isso permite novas investigações sobre práticas ágeis em diferentes contextos e contribuindo para a diversificação das fontes de dados disponíveis para a comunidade de pesquisa.

7. Considerações finais

Este conjunto de dados minerado do *GitLab* e disponibilizado em formato CSV, tem como objetivo auxiliar na educação e pesquisa sobre desenvolvimento ágil de software. Foi criado com a finalidade de treinamento e pesquisa em Estimativa de *user stories* em *story points*, mas também contém informações relevantes para outros aspectos da engenharia de *software*. Além disso, permite a replicação de descobertas de estudos anteriores.

Como trabalho futuro, sugerimos a disponibilização do conjunto de dados em diferentes formatos, a fim de aumentar sua acessibilidade e adaptabilidade a diferentes contextos de pesquisa e desenvolvimento ágil de software. Outra sugestão é disponibilizar o conjunto de dados no formato NoSQL.

Outra sugestão de trabalho futuro é converter os dados em *scripts* SQL, permitindo a sua utilização em modelos relacionais de bancos de dados. Adicionalmente, outra sugestão é oferecer um contêiner *Docker* com um Sistemas de Gerenciamento de Banco de Dados configurado, visando simplificar o uso e a adoção do conjunto de dados por parte de outros pesquisadores.

Referências

BECK, K. Extreme Programming Explained: Embrace Change. 1ª ed. Boston: Addison-Wesley, 2001.

CHAPARRO, O., LU, J., ZAMPETTI, F., MORENO, L., DI PENTA, M., MARCUS, A., BAVOTA, G., AND NG, V. Detecting missing information in bug descriptions. **Proceedings of the ACM SIGSOFT Symposium on the Foundations of Software Engineering** Part F130154 (2017), 396–407.

CHOETKIERTIKUL, M., DAM, H. K., TRAN, T., GHOSE, A., AND GRUNDY, J. Predicting Delivery Capability in Iterative Software Development. **IEEE Transactions on Software Engineering** 44, 6 (2018), 551–573.

CHOU, P., CROWSTON, K., DAHLANDER, L., MINERVINI, M. S., AND RAGHURAM, S. GitLab: work where you want, when you want. Journal of Organization Design 9, 1 (2020).

COHN, M. **User Stories Applied: For Agile Software Development**. 1ª ed. Boston: Addison-Wesley, 2005.

DALPIAZ, F. 2018. Disponível em: https://data.mendeley.com/datasets/7zbk8zsd8y.

DIMÍTRIJEVIC, S., JOVANOVIC, J., AND DEVEDZIC, V. A comparative study of software tools for user story management. **Information and Software Technology** 57 (2015), 352–368.

DRAGICEVIC, S., CELAR, S., AND TURIC, M. Bayesian network model for task effort estimation in agile software development. **Journal of Systems and Software** 127 (2017), 109–119.

DYBÅ, T., AND DINGSØYR, T. Empirical studies of agile software development: A systematic review. **Information and Software Technology** 50, 9-10 (2008), 833–859.

GAVIDIA-CALDERON, C., SARRO, F., HARMAN, M., AND BARR, E. T. The Assessor's Dilemma: Improving Bug Repair via Empirical Game Theory. **IEEE Transactions on Software Engineering** 47, 10 (2021), 2143–2161.

HARDT, M.; NARAYANAN, A. **Data and Society: A Critical Introduction**. Cambridge: Cambridge University Press, 2019.

HUANG, Y., WANG, J., WANG, S., LIU, Z., WANG, D., AND WANG, Q. Characterizing and predicting good first issues. International Symposium on Empirical Software Engineering and Measurement (2021).

JADHAV, D., KUNDALE, J., BHAGWAT, S., AND JOSHI, J. A Systematic Review of the Tools and Techniques in Distributed Agile Software Development. Agile Software Development: Trends, Challenges and Applications (2023), 161–186.

JIMÉNEZ, S., ALANIS, A., BELTRÁN, C., JUÁREZ-RAMÍREZ, R., RAMÍREZ-NORIEGA, A., AND TONA, C. Usqa: A user story quality analyzer prototype for supporting software engineering students. **Computer Applications in Engineering Education** (2023).

JUST, R., JALALI, D., AND ERNST, M. D. Defects4J: A database of existing faults to enable controlled testing studies for Java programs. **2014 International Symposium on Software Testing and Analysis**, ISSTA 2014 - Proceedings (2014), 437–440.

KIM, D.; PARK, S. Structured versus unstructured data: Implications for data sharing among corporations. **Data Science Review**, v. 5, n. 2, p. 125-138, 2020.

KONONOV, Dmitriy. ICT for my work: **The GitLab – is it a more powerful alternative to GitHub**?. 2018. Disponível em: https://kononovdm.github.io/2018/11/GitLab-review/.

MAYER-SCHÖNBERGER, V.; CUKIER, K. **Big Data: A Revolution That Will Transform How We Live, Work, and Think.** New York: Houghton Mifflin Harcourt, 2013.

MANI, S., SANKARAN, A., AND ARALIKATTE, R. Deeptriage: Exploring the effectiveness of deep learning for bug triaging. **ACM International Conference Proceeding Series** (2019), 171–179.

MERGEL, I. **Agile innovation management in government: A research agenda**. Government Information Quarterly 33, 3 (2016), 516–523.

MOUNTAIN SOFTWARE, G., 2004. Disponível em https://www.mountaingoatsoftware.com/uploads/documents/example-user-stories.pdf.

MURPHY, Chris et al. GitLab **CI/CD and DevOps: Reference Design and Implementation Guide for GitLab Pipelines**. 2020. Disponível em: https://docs.gitlab.com/ee/ci/examples/..

NEO, Giseldo et al. User Story Tutor (UST) to Support Agile Software Developers. In: **CSEDU (2)**. 2024. p. 51-62.

ORTU, M., DESTEFANIS, G., ADAMS, B., MURGIA, A., MARCHESI, M., AND TONELLI, R. The JIRA repository dataset: Understanding social aspects of software development. **ACM International Conference Proceeding Series** 2015-October (2015).

ORTU, M., MURGIA, A., DESTEFANIS, G., TOURANI, P., TONELLI, R., MARCHESI, M., AND ADAMS, B. The emotional side of software developers in JIRA. **Proceedings - 13th Working Conference on Mining Software Repositories**, MSR 2016 (2016), 480–483.

PANDAS Documentation. **IO Tools** (Text, CSV, HDF5, …). Disponível em: https://pandas.pydata.org/pandas-docs/stable/reference/io.html.

PMI. Success Rates Rise - **2017 9th Global Project Management Survey**. Tech. rep., PMI, 2017.

PORRU, S., MURGIA, A., DEMEYER, S., MARCHESI, M., AND TONELLI, R. Estimating story points from issue reports. **ACM International Conference Proceeding Series** (2016).

RIGBY, D. K., SUTHERLAND, J., AND NOBLE, A. Agile Scale: How to go from teams to hundreds. **Havard Business Review** May-June, June (2018), 1–3.

SABBAGH, R. Scrum: Gestão ágil para projetos de sucesso. Editora Casa do Código, 2014.

SCHWABER, K.; SUTHERLAND, J. **The Scrum Guide**. 2020. Disponível em: https://scrumguides.org.

SMITH, J. Privacy concerns in the sharing of corporate data for research purposes. **Corporate Data Journal**, v. 8, n. 1, p. 112-126, 2017.

SOARES, R. G. Effort Estimation via Text Classification And Autoencoders. In 2018 International Joint Conference on Neural Networks (IJCNN) (2018), vol. July, **IEEE**, pp. 1–8.

SUTHERLAND, J. **SCRUM: A arte de fazer o dobro de trabalho na metade do tempo**. Leya, 2014.

TAWOSI, V., AL-SUBAIHIN, A., MOUSSA, R., AND SARRO, F. Agile effort estimation: Have we solved the problem yet? insights from a replication study. **IEEE Transactions on Software Engineering** 49, 4 (2022), 2677–2697.

TAWOSI, V., AL-SUBAIHIN, A., MOUSSA, R., AND SARRO, F. A Versatile Dataset of Agile Open Source Software Projects. In **Proceedings - 2022 Mining Software Repositories Conference**, MSR 2022 (2022), pp. 707–711.

TAWOSI, V., SARRO, F., PETROZZIELLO, A., AND HARMAN, M. Multi-Objective Software Effort Estimation: A Replication Study**. IEEE Transactions on Software Engineering** 48, 8 (2022), 3185–3205.

TAWOSI, V., MOUSSA, R., AND SARRO, F. Investigating the Effectiveness of Clustering for Story Point Estimation. In Proceedings - 2022 **IEEE International Conference on Software Analysis, Evolution and Reengineering**, SANER 2022 (2022), pp. 827–838.

TOMASSI, D. A., DMEIRI, N., WANG, Y., BHOWMICK, A., LIU, Y. C., DEVANBU,P. T., VASILESCU, B., AND RUBIO-GONZALEZ, C. BugSwarm: Mining and Continuously Growing a Dataset of Reproducible Failures and Fixes. Proceedings - **International Conference on Software Engineering** 2019-May (2019), 339–349.

TRIMBLE, J., SHIRLEY, M. H., AND HOBART, S. G. Agile: From software to mission system. **14th International Conference on Space Operations**, 2016 (2016), 1–8.

UMER, Q., LIU, H., AND ILLAHI, I. CNN-Based Automatic Prioritization of Bug Reports. **IEEE Transactions on Reliability** 69, 4 (2020), 1341–1354.

VALDEZ, A., OKTABA, H., GOMEZ, H., AND VIZCAINO, A. Sentiment analysis in jira software repositories. **Proceedings - 2020 8th Edition of the International Conference in Software Engineering Research and Innovation**, CONISOFT 2020 (2020), 254–259.

VENKATRAMAN, K.; AKASHVARMA, M.; SIDDHARTH, S. Enhancing Software Test Effort Estimation using Ensemble Learning Algorithms. In: **2023 4th International Conference on Intelligent Technologies (CONIT)**. IEEE, 2024. p. 1-5.

WIKIPEDIA. CSV. Disponível em: https://pt.wikipedia.org/wiki/Comma-separated_values.

Apêndice E

User Story Tutor to Support Agile Software Developers

Artigo publicado na “CSEDU 2024 - International Conference on Computer Supported Education” com o título “User Story Tutor (UST) to Support Agile Software Developers”. Além disso, recebeu o prêmio de melhor artigo de estudante no mesmo evento [94]. Sendo posteriormente encaminhado para publicação no Journal Springer Nature Computer Science [29].

User Story Tutor (UST) to Support Agile Software Developers

Giseldo da Silva Neo[1][a], José Antão Beltrão Moura[2][b], Hyggo Almeida [2][c], Alana Viana Borges da Silva Neo [2][d], and Olival de Gusmão Freitas Júnior[3][e]

[1]*Campus Viçosa, Federal Institute of Alagoas, Viçosa, Brazil*

[2]*Center for Electrical and Computer Engineering, Federal University of Campina Grande, Campina Grande, Brazil*

[3]*Institute of Computing, Federal University of Alagoas, Maceió, Brazil*

giseldo.neo@ifal.edu.br, {antao,almeida}@computacao.ufcg.edu.br, alana.neo@copin.ufcg.edu.br, olival@ic.ufal.br

Keywords: User Story, Story Points, Education, Recommendation, Readability

Abstract: User Stories record what must be built in projects that use agile practices. User Stories serve both to estimate effort, generally measured in Story Points, and to plan what should be done in a Sprint. Therefore, it is essential to train software engineers on how to create simple, easily readable, and comprehensive User Stories. For that reason, we designed, implemented, applied, and evaluated a web application called User Story Tutor (UST). UST checks the description of a given User Story for readability, and if needed, recommends appropriate practices for improvement. UST also estimates a User Story effort in Story Points using Machine Learning techniques. As such UST may support the continuing education of agile development teams when writing and reviewing User Stories. UST's ease of use was evaluated by 40 agile practitioners according to the Technology Acceptance Model (TAM) and AttrakDiff. The TAM evaluation averages were good in almost all considered variables. Application of the AttrakDiff evaluation framework produced similar good results. Apparently, UST can be used with good reliability. Applying UST to assist in the construction of User Stories is a viable technique that, at the very least, can be used by agile developments to complement and enhance current User Story creation.

1 INTRODUCTION

Every year, a report that consolidates data from technology projects from different companies in several countries is published on the internet by the Standish Group International Organization. The report is called Chaos Report (StandishGroup, 2015). This report indicates that less than 1/3 of the surveyed software technology projects were completely successful. Most of the projects did not reach completion within the planned time and cost budget estimated.

Software development is a complex process that involves many variables and is prone to several failures (Sommerville, 2011). Much of this failure is related to the specification of what should be done and other factors. To minimize this problem some methodologies and frameworks, like Agile methods, can provide a conceptual structure for conducting this software engineering projects.

Agile can be understood as a set of behaviors, processes, practices, and tools used to create products and subsequently make them available to end users (Cohn, 2005). One of the best-known representatives of Agile methods is SCRUM (Sutherland, 2014). It focuses primarily on the aspect of what must be done. In SCRUM requirements must be specified at an adequate level of clarity, neither complex nor too rigid. An important part of this method is writing, interpreting, and implementing what is called a User Story.

A User Story is a short and simple sentence about a feature (written from the perspective of the user who wants it) and is used to define the scope of a software project (Cohn, 2004). It is a requirements analysis technique that captures the "who", "what" and "why" concisely and simply, usually limited in detail, so that it can be written by hand on a small note card of paper. These User Stories are generally stored in software that manages all the project life-cycle (Jadhav et al., 2023). By analyzing these raw data, recorded in these tools, we can extract information for various software

[a] https://orcid.org/0000-0001-5574-9260
[b] https://orcid.org/0000-0002-6393-5722
[c] https://orcid.org/0000-0002-2808-8169
[d] https://orcid.org/0009-0000-1910-1598
[e] https://orcid.org/0000-0003-4418-8386

engineering research (Tawosi et al., 2022a).

However, writing a good User Story can be difficult. The User Story can be very shallow and not present adequate detail to understand the expected final result, or, conversely, it can be too comprehensive. For example: A stakeholder may confuse the level of detail of a User Story and write the scope of an entire module or system, which is not appropriate. Also, the quality of User Stories can have a big impact when the agile team makes estimates. The writing of good user stories, which can be used to estimate effort, appears as one of the 5 most important agile problems informed by 119 developers (Andrade, 2021).

Improving the creation of the User Story is crucial for better planning and consequently the success of the project. Therefore, the objective of this study is to assist the User Story creation process, recommending improvements, using natural language processing in an intelligent learning environment, i.e., a pedagogical agent that assists in the learning process, characterizing itself as a tutor of content or more adapted strategies. The hypothesis is that this environment can help agile teams build better user stories. An expected contribution is to help development teams that use agile practices to build better user stories.

The tool User Story Tutor (aka UST), proposed in this article, receives as input a User Story text in English and presents personalized recommendations for improving it, with the support of a large language model (LLM). LLMs are very large deep learning models that are pre-trained on vast amounts of data. The tool also presents a prediction in Story Points, generated by a machine learning algorithm trained with data from other projects. The user is also presented with the User Story's readability indexes, as they can be used to represent an indicator of text clarity.

We use the Design Science Research search methodology with 3 phases: problem identification, solution design, and evaluation to build the proposed tool (Wieringa, 2014). For evaluation, a survey was carried out with the support of the Technology Acceptance Model (TAM) framework (Davis et al., 1989) and the AttrakDiff evaluation framework (Hassenzahl et al., 2003). 40 agile practitioners who did not engage in UST's development, evaluated the proposal solution and responded to the Survey. The TAM and AttrakDiff evaluation results indicate that UST meets the established objectives, with good acceptance from participants.

The remainder of the paper is organized as follows. Section 2 summarizes the main concepts that facilitate the understanding of subsequent sections. Section 3 discusses the methodology and methodological artifacts adopted in UST's R&D. Section 5 brings more technical details of UST. Section 4 addresses UST's evaluation results. Section 6 is devoted to related work. Section 7 explores threats to the validity of our investigation and its results. Lastly, Section 8 offers the final considerations.

2 BACKGROUND

2.1 User Stories

User Stories are a central piece in the development of requirements for teams that use agile development. A characteristic of agile methods is their focus on fast, value-added deliveries in short periods, dealing with changes as quickly as possible (Dybå and Dingsøyr, 2008). This approach has shown results and has been used in project management (PMI, 2017), both in industry (Trimble et al., 2016; Rigby et al., 2018) and in government (Mergel, 2016).

SCRUM is an agile method based on fixed time cycles called sprints, where teams work to achieve well-defined objectives; these objectives are represented in the Product Backlog, a list of things to do that is constantly updated and re-prioritized (Sutherland, 2014). Software requirements are usually stored in User Stories. These artifacts describe the activities that the development team will estimate and build and are written in natural language.

Most teams that use User Stories, or even the agile method, also use software tools to manage the project and mainly to keep a record of their user stories (Jadhav et al., 2023). By analyzing the data recorded by these tools we can extract information for various software engineering research, including research on how to improve these same User Stories (Jiménez et al., 2023).

GitLab is one of these management tools used by agile teams to record User Stories (Choudhury et al., 2020). It allows software engineers to automate many actions during the development cycle, including recording and changing User Stories (Dimitrijević et al., 2015). In GitLab, the User Story is registered as an Issue, and for each User Story various pieces of information are stored, such as the title, the task description, and its estimate in Story Points.

This data stored in these management tools can support decision-making in various software engineering scenarios, such as: assigning User Stories (Mani et al., 2019), improving the description of User Stories (Chaparro et al., 2017), iteration planning (Choetkiertikul et al., 2018), sentiment analysis of developers who write User Stories (Ortu et al.,

2015; Ortu et al., 2016; Valdez et al., 2020), effort estimation of User Stories (Porru et al., 2016; Soares, 2018; Dragicevic et al., 2017; Choetkiertikul et al., 2019; Tawosi et al., 2022b), and prioritization of User Stories (Gavidia-Calderon et al., 2021; Huang et al., 2021; Umer et al., 2020).

2.2 Recommendation Systems

Recommendation systems are software tools and techniques that provide suggestions for items that are more likely to arouse the interest of a particular user (Ricci et al., 2010). Recommendations are important for the learning process by allowing teachers and students to find content more appropriately, according to their profile and needs.

The evolution of Recommendation Systems is moving towards a set of hybrid techniques, which combine two or more different Recommendation techniques, to resolve the limitations and obtain the advantages of each of them (Bobadilla et al., 2013). A hybrid Recommendation system is a term used to describe any Recommendation system that combines several Recommendation techniques to produce an output. Burke (2007) cites the following types: Collaborative, Content-based, demographic, and knowledge-based (Burke, 2007).

Recommendation systems have gained a lot of popularity in the educational field, generating various types of recommendations for students, teachers, and schools. They can reduce student information overload by recommending the "right" information at the right time and in the right format of interest to the student (Odilinye and Popowich, 2020).

More recently, LLM-type models (for example, ChatGPT from OpenAI) can be also used for recommendations. OpenAI is the company behind an LLM called ChatGPT. They offer this large-scale multimodal model to be used for third parties in situations where reliability is not critical (OpenAI, 2023).

Despite some problems with the LLM recommendation approach: responses from an LLM are not completely reliable, have a limited context window, and do not learn (OpenAI, 2023). They can be useful in specific domains. They are suitable in situations where there are well-defined intentions, for example, opening a bank account, scheduling an air ticket, or improving a User Story, given their controlled structure and predictable output. In this study, OpenAI will be used as a synonym for OpenAI API, or even ChatGPT, and in the context of this paper, a recommendation is a text to improve the User Story, which the system suggests to the user.

2.3 Readability Indexes

There are some readability indexes commonly used in the literature to predict the reading ease of the text. They are used to determine the readability of an English passage and they are already used in fake news and opinion spam detection. This section describes the 4 most common of them.

Readability indexes can be interpreted as a numerical indicator of how much easier it is for other people to read writing text (DuBay, 2004). To extract this numerical information some of the algorithms use the count of words, characters, sentences, syllables, and a list of complex words in their formula.

Readability indexes have been used by educators since 1920. In 1980 there were already 200 known calculation formulas (DuBay, 2004). They have already been criticized by researchers, who point out their limitations (Koenke, 1971). However, empirical experiments confirmed the relationship between these indexes and the readability of the text (Bogert, 1985).

Gunning's Fog Index is the most frequently used and studied index and has been extensively used to analyze text (Bogert, 1985). It is a numerical number assigned to a given text that uses words, sentences, and a list of complex words in their formula. The higher the value, the more complex the text. It was created by Robert Gunner in 1954. The Greater the percentage of complex words, the harder the text is to read. The Higher the index, the lesser the readability. His algorithm and method of calculation are well documented (Gross and Sadowski, 1985). It can be computed by adding the average sentence length and the percent of complex words (words of three or more syllables) and multiplying that sum by 0.4. Like in the formula presented in Equation 1.

$$0.4 \cdot \left[\left(\frac{words}{\text{sentences}} \right) + 100 \cdot \left(\frac{words\ complex}{words} \right) \right] \quad (1)$$

Another index of text readability is Flesch Reading Ease, according to (Textstat, 2023). The higher the value, the more difficult it is to read the text. Its maximum value is 121.22. There is no minimum value, negative scores are also valid. Equation 2 presents the calculation of the Flesch Reading Ease. It is one of the oldest and most widely used tests and is only dependent on two factors: The Greater the average sentence length, the harder the text is to read. The greater the average number of syllables in a word, the harder the text is to read. The higher the score, the greater the readability.

$$206.835 - 1.015 \cdot \left(\frac{words}{sentences} \right) - 84.6 \cdot \left(\frac{syllables}{words} \right) \quad (2)$$

Coleman Liau Index is another complexity index (Textstat, 2023), but this time using another Equation 3. Where L is the average number of letters per 100 words, and S is the average number of sentences per 100 words.

$$CLI = 0.0588 \cdot L - 0.296 \cdot S - 15.8 \quad (3)$$

Finally, the Automated Readability Index is calculated from the following Equation 4 (Textstat, 2023).

$$4.71 \cdot \left(\frac{characters}{words} \right) + 0.5 \cdot \left(\frac{words}{sentences} \right) - 21.43 \quad (4)$$

3 METHODOLOGY

The research methodology used in this research was Design Science Research (Wieringa, 2014). We designed a web application called User Story Tutor (UST) that uses Natural Language Processing, Readability Indexes and Machine Learning Prediction as a proof of concept to improve User Story writing. We used a survey, supported by a questionnaire in Google Forms, to evaluate UST. The development was carried out in the following stages:

- Literature review;
- Design of a predictive model that predicts the number of Story Points from the User Story using Machine Learning;
- Definition of the basic text readability indexes that can be extracted from User Stories;
- Design of a recommendation module via querying the OpenAI API;
- Implementation all 3 modules as a web application;
- Internal evaluation using many User Stories from real projects;
- Evaluation with participants in a Survey;
- Qualitative and quantitative analysis of the Survey results.

To evaluate UST, we carried out a survey based on the Technology Acceptance Model (TAM) framework (Davis et al., 1989) and AttrakDiff evaluation framework (Hassenzahl et al., 2003). TAM is an information systems theory that models how users accept and use technology - please see Figure 1. For the TAM statistical test, Cronbach's alpha was used. The AttrakDiff (Hassenzahl et al., 2003) test presents quality factors (hedonic and pragmatic) that can help to better evaluate the proposal, complementing the TAM framework. The Survey collected participants' perceptions and suggestions regarding UST, whose objective is to assist agile practitioners in building better User Stories. The questions used in the survey are presented in Section 5.

UST's architecture was divided into 3 modules (Recommender, Estimator, and Readability) and for each module, certain procedures were selected and executed. These selected general procedures will be detailed in the following paragraphs.

The Recommender module is responsible for recommending improvements to User Stories by returning text in natural language. The LLM model used (made available by OpenAI) is already trained with a large amount of text, extracted from the entire internet (Torrent et al., 2023). However, customization of the prompt is necessary to better personalize the return.

To customize the response from the OpenAI LLM model, we send a prompt to configure the return according to the needs of the recommendation system. Following OpenAI's guidelines for building effective prompts, 3 main recommended techniques were used in the prompt design.

1. Clarity in instruction

   We seek a clear and precise prompt to not generate doubts when returning the recommendation. The probability of a good return recommendation depends on the objectivity of the hidden prompt sent along with the recommendation.

2. Split complex tasks into simpler tasks

   Intending to limit the task, we sent (a prompt) text to limit the set of return possibilities, as complex tasks generally have a higher error rate than simpler task requests;

3. Test changes

   Several interactions were necessary to create the prompt used in search of improvements and following the good practices recommended by OpenAI itself.

The Estimator module uses a supervised learning algorithm, selected according to its best prediction capacity to predict the Story Points from the informed User Story text. To elaborate the predictor module we followed the techniques: data collection; data exploration; data preparation; creation, training, and validation; adjustment of hyperparameters, and implementation of the model.

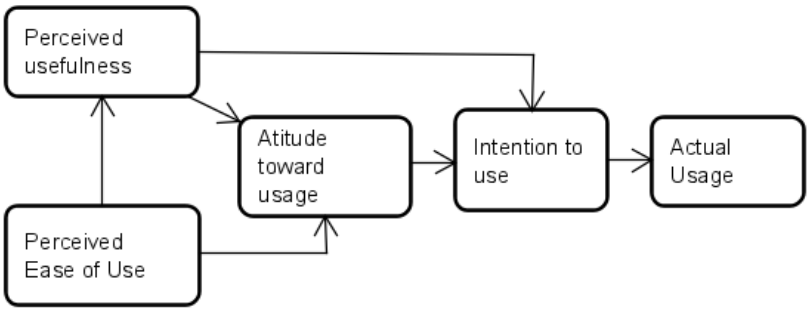


Figure 1: TAM. Adapted from (Alomary and Woolard, 2015)

The User Story Estimator predictor was trained with other User Stories from other agile projects. This dataset was collected from real open-source projects extracted from an open-source repository. More details of the dataset are presented in Section 4.1. The metric used to select the best algorithm and for hyperparameters adjusting was the Mean Absolute Error with cross-validation. In the end, The predictor model was trained with all the data and made available for the proposed application.

Finally, the module related to Readability presents 4 text readability indexes: Gunning Fog, Automated readability index, Coleman Liau Index, and Flesch Reading Ease (please refer to Section 2). But it is important to highlight that for this proposal, to facilitate the interpretation of the readability index in general, a variable called Final Result was also created, which is the arithmetic mean of the 4 selected Indexes.

4 THE USER STORY TUTOR

This section presents an overview of the technologies used to build the User Story Tutor (aka UST), its high-level architecture, the dataset used, and its application interface.

Our idea is that UST supports agile teams during the construction of User Stories and assists the development process during the User Story preparation phase and in estimating task effort. UST consists basically of a web application that can be accessed using a browser on mobile devices, PCs, and notebooks. UST uses an LLM provided by OpenIA to recommend improvements and present readability indexes. The main parts of the tool are discussed below.

The application was designed around 3 modules with well-defined functions. The Recommender module responds to User Story recommender requests. It is responsible for maintaining the prompt and combining it with the text of the new User Story, querying OpenAI via API, and preparing the return for presentation to the user. The module that performs User Story estimates in Story Points uses a predictive model, already trained with historical data to assist developers in their estimates, acting as a reference for the team responsible for estimating effort. Readability indexes are extracted from text with basic natural language processing techniques. An image of the architecture is presented in Figure 2.

For coding, we used StreamLit [1] - a library for building open-source applications for machine learning and data science. Python [2] was used as the language - a programming language that has been increasing its market share, mainly in applications that use machine learning. The Recommender Module performs a query to OpenAI. The scikit-learn libraries were also used [3]. All source-code of the project was available at Github [4]. The UST can be tested at StreamLit Cloud [5].

4.1 Dataset

We have made a new dataset (aka NEODATASET) available together with UST. This dataset encompasses data from 34 software development projects, with 40.014 User Stories taken from GitLab repositories, totaling 163.897 Story Points. It is made available on GitHub [6] so that the entire interested community can contribute, similarly to what happens with other datasets.

This dataset was mined during January 2023 and April 2023. The mining process targeted GitLab's top open-source projects. The selected projects employ agile software development methodologies and had the size of their tasks recorded in *Story Points*. To mine information from GitLab, we created an ex-

[1] https://streamlit.io
[2] https://www.python.org/
[3] https://scikit-learn.org
[4] https://github.com/giseldo/userstory
[5] https://userstoryteach.streamlit.app
[6] https://github.com/giseldo/neodataset

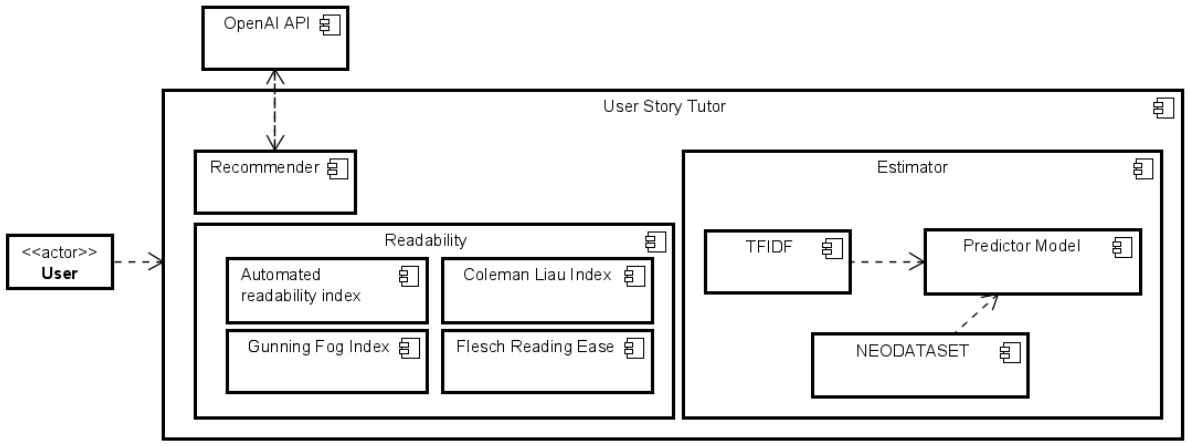


Figure 2: UST Architecture

traction tool implemented in Python that connects to GitLab via API

Only Tasks with the State attribute equal to Closed and that have the *weight* attribute filled in were collected. The *weight* field is used in GitLab to record the effort in Story Points. More information about the projects included in the dataset is also available directly from GitLab.

The projects in the dataset have different characteristics and cover different programming languages, different business domains, and different geographic locations of the team. The main entity of the dataset is the User Story (or Issue), which contains the main information. The dataset has more than 70 attributes and is stored in JSON and CSV format, given the simplicity of dealing with both formats.

The dataset presented here includes projects which were not used by previous studies. There are already previous studies that extracted data from the Jira management tool to build predictive models (Tawosi et al., 2022a; Choetkiertikul et al., 2019; Porru et al., 2016; Scott and Pfahl, 2018), but projects extracted from GitLab are rarer.

Just as (Tawosi et al., 2022a) did, we are sharing all the data collected. Therefore, the most common thing is to share only the data from the dataset considered in the study itself, as done, for example in (Choetkiertikul et al., 2019), and not all the data collected.

The expected contribution is that this data set can assist education and research on agile software development. Although our dataset was initially designed for Story Points and User Story estimation training and research, it also includes information relevant to other software engineering aspects. In addition to providing a possibility to reproduce findings from other studies.

4.2 User Interface

The first screen of the UST (Figure 3) is where a developer from the agile team informs their User Story. Any User Story from any real project can be used. Then, after the developer enters the description of the User Story in text format, he clicks on the "Analyze" button. The UST then initializes the necessary threads that trigger the responses of the existing modules. The language of the UST interface was English.

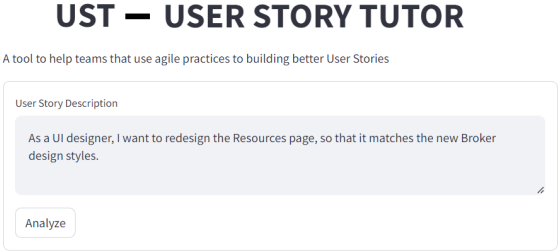


Figure 3: Home Screen

The recommender module is a module that returns suggestions from the developer Story description text entered by the user (Figure 4). The model provided by OpenAI was used, specifically the gpt-3.5-turbo. The parameters used by the recommender module are sent via a hidden prompt, and they are presented in Table 1. They followed a refinement process similar to creating a search key in a systematic literature review, being revised and adapted until we reached the final version presented with the support of the produced data set in Section 4.1.

For the Recommender module, a valid input from the user would be this imagined User Story: *"As a UI designer, I want to redesign the Resources page, so that it matches the new Broker design styles."* The Recommendation text that UST would return to the

Table 1: OpenAI personalized prompt

role	content
system	You are a scrum master, skilled in creating better User Stories for agile software projects
user	How can I improve this user story: + *User Story text informed by user in english*

user is: *" To improve this user story, you can consider adding more details and making the objective clearer. Here's an improved version of the user story: As a UI designer, I want to redesign the Resources page to improve the user experience and align it with the new Broker design styles, enhancing the overall consistency and visual appeal of the application. Additionally, you can further refine the user story by specifying the specific changes or improvements you plan to make to the Resources page".*

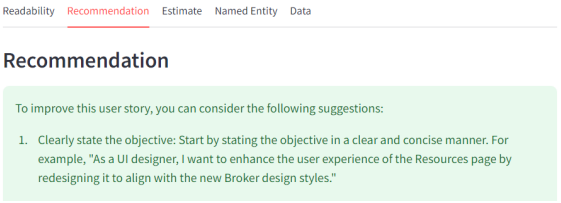


Figure 4: Recommendation Module Example

For the User Story readability module (Figure 5), the readability indexes of the text are extracted using the textdescriptive library (available in Python) [7] and presented on the screen with the "metrics" component from StreamLit in the first tab. The purpose of the readability module is to allow the creator of the User Story to see some quantitative measure of how easy the text of their User Story is to read.

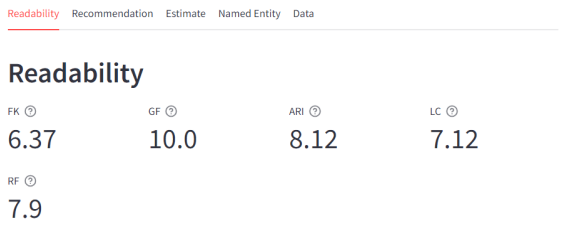


Figure 5: Readability Indexes Module

Finally, the Effort Estimation module (Figure 6) performs an effort estimate in Story Points based on the User Story description. The predictive model and vectorizer used are loaded with the Joblib library. The selected algorithm was SVM. The User Story text is transformed into a bag-of-words using the TFIDF technique. In production, both the vectorizer and the model are loaded into memory for prediction. After the User Story text is transformed into a matrix with the vectorizer, it is passed on to the model predictor, which returns the estimate in Story Points. The loaded Model was previously trained with data from NEODATASET.

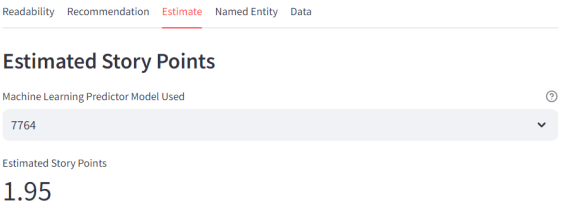


Figure 6: Estimation Module

5 EVALUATION

This section presents a qualitative and quantitative evaluation of the tool with the support of the TAM framework and AttrakDiff and discusses the results. The survey was conducted in December 2023 with an online questionnaire in Google Forms. The questionnaire was first examined for comprehensiveness, quality and adequacy to the investigation at hand by a panel of 6 experts who had amongst them 7 years of experience in agile development. The questionnaire used a 5 level Likert scale to gauge the respondent's agreement (from none or level 1), passing through little (2), neutral (3), somewhat (4) to full (level 5) with statements made concerning UST. The experts' comments and suggestions led to the adjustment of the questionnaire which was then applied to a sample of respondents.

5.1 Sample Characterization

Our sample of survey respondents is made up of 40 Brazilian participants. 70% of those who responded to the survey had worked directly with agile methodologies. More than half of these (56%) had worked in a software factory and had already worked as a member of a software development team. Of these, 20% have been Scrum Masters and 10% have been Product Owners. The other participants had generally participated in academic activities related to software engineering. On average, our sample was made up of professionals with 3 years of experience in agile methodology.

[7] https://pypi.org/project/textdescriptives

5.2 TAM

Each of the 4 constructs of the Technology Assessment Model is analyzed below: perception of usability, perception of ease of use, external variables, and attitude.

The perception of usability is the level at which a person believes that using UST improves the performance of their tasks. To analyze the perception of usability, (Table 2) the mean, median, and standard deviation of the Likert scale responses were analyzed. If the Mean or Median is above the threshold which we chose to be "3" (neutral) in our experiments, this possibly indicates that the participants have a positive attitude towards the perception of using the tool (Dantas et al., 2019).

Evaluating the responses, it is possible to infer that participants generally have a positive attitude toward the perceived usability of the tool (Table 2).

Table 2: Perception of usability

Definition	Mean	Med	SD
V1 Using the tool is useful to improve my User Stories	4.45	5.00	0.80
V2 I learned how to build better User Stories after using the tool	4.07	4.00	1.10

The perception of ease of use is the Level at which the person presents their perception of the tool in terms of ease of learning and operation. Table 3 describes the mean, median, and standard deviation of the responses related to perceived ease of use. All averages are above the adopted threshold, therefore also regarding good perception and ease of use of UST. A standard deviation above one indicates a high dispersion in responses.

Table 3: Perceived ease of use

Definition	Mean	Med	SD
V3 Learning to use the tool was easy for me	3.70	4.00	1.28
V4 Searching for information in this tool was simple	3.72	4.00	1.09
V5 Accessing the tool is simple	4.07	4.00	1.14

An analysis of external variables, which provides a better understanding of what influences perceived utility and ease of use, is presented in Table 4. A median above 4 is a good indicator that the external characteristics were well accepted by users.

Table 4: External variables

Definition	Mean	Med	SD
V6 The application's navigation attributes - menu, icons, links, and buttons - are clear and easy to find	4.02	4.00	1.17
V7 The tool has a good interface	3.9	4.00	1.20

The data characterized as Attitude, which is the Intention of the individual to use the tool, are presented in Table 5. In the same way, as with the other constructs, we have a mean above the threshold.

Table 5: Attitude

Definition	Mean	Med	SD
V8 I believe it is better to use the tool to help create the user story than not to use it.	4.25	4.00	0.88
V9 I intend to use the tool to create better user stories and to plan my tasks better	3.85	4.00	1.15

For statistical confirmation, Cronbach's (Gliem and Gliem, 2003) test was used for the Likert scale questionnaire, the same technique used by (Dantas et al., 2019). Cronbach's Alpha is a way to measure the internal consistency of a questionnaire or survey. Cronbach's Alpha ranges between 0 and 1, with higher values indicating that the survey or questionnaire is more reliable. An interpretation of Cronbach's alpha is presented in Table 6.

Table 6: Internal consistency from the Survey. Adapted from (Zach, 2023)

Cronbach Alpha	Internal consistency
$0.9 \leq \alpha$	Excellent
$0.8 \leq \alpha < 0.9$	Good
$0.7 \leq \alpha < 0.8$	Acceptable
$0.6 \leq \alpha < 0.7$	Questionable
$0.5 \leq \alpha < 0.6$	Poor
$\alpha < 0.5$	Unacceptable

A limit adopted in this research is the Cronbach alpha index greater than 0.7 for the variables analyzed and with confidence in the 95% range. From the reported values, as shown in Table 7, we understand that almost all constructs analyzed are above the established limit. This leads to the conclusion that the

internal consistency of this survey is acceptable.

Table 7: Cronbach of TAM constructs

constructs	Cronbach	IC
Usability	0.81	[0.64 0.90]
Ease of use	0.92	[0.87 0.95]
External variables	0.87	[0.77 0.93]
Attitude	0.73	[0.49 0.85]

5.3 AttrakDiff

In Figure 7 we present the portfolio of results of the AttrakDiff test (Hassenzahl et al., 2003). This test presents factors that can help better evaluate the proposal, complementing what the TAM framework presents. The AttrakDiff short test type was used, which presents 10 questions to users and infers the metrics reported below.

In Figure 7 the vertical axis of the portfolio view displays the hedonic quality (bottom = low extent). The horizontal axis shows the pragmatic quality (left = low extent). Depending on the dimensions values the product will lie in one or more character regions. The bigger the confidence rectangle, the less sure one can be about which region it belongs. A small confidence rectangle is an advantage because it means that the investigation results are more reliable and less coincidental. The confidence rectangle shows if the users are at one in their evaluation of the product. The bigger the confidence rectangle, the more variable the evaluation ratings (Hassenzahl et al., 2003). So, the answers point to a small trust rectangle in the upper right quadrant, as a task-oriented tool.

In Figure 8 we present the diagram of average values. The average values of the AttrakDiff dimensions for the evaluated product are plotted on the diagram. In this presentation, hedonic quality distinguishes between the aspects of stimulation and identity. Furthermore, the rating of attractiveness is presented (Hassenzahl et al., 2003).

In Figure 9 we present the description of word pairs. The mean values of the word pairs are presented here. Of particular interest are the extreme values. These show which characteristics are particularly critical or particularly well resolved (Hassenzahl et al., 2003). Better results are placed in the positive quadrant, which can be inferred from the consolidated results in Figure 9. Almost all items evaluated were in the positive quadrant, except the pair (cheap-premium).

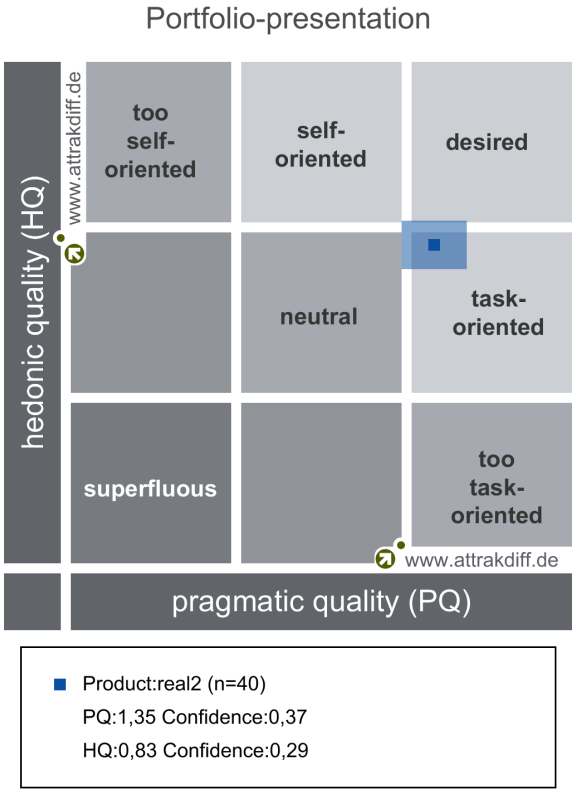


Figure 7: Portfolio of results

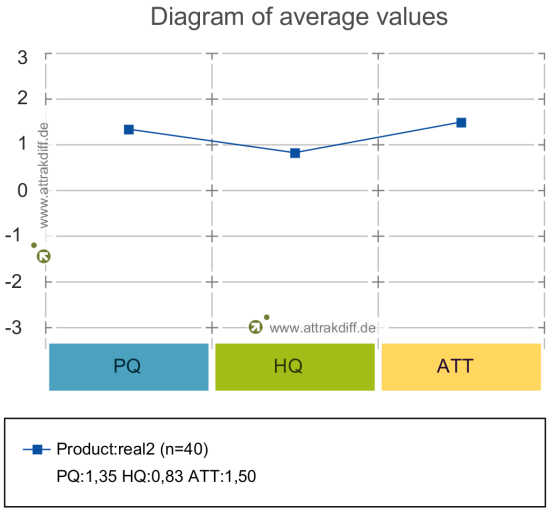


Figure 8: Diagram of average values

6 RELATED WORK

Improving the quality of user stories is a line of research that is gaining momentum due to advances in artificial intelligence. Generally, the most classic approaches use the transformation to an intermediate

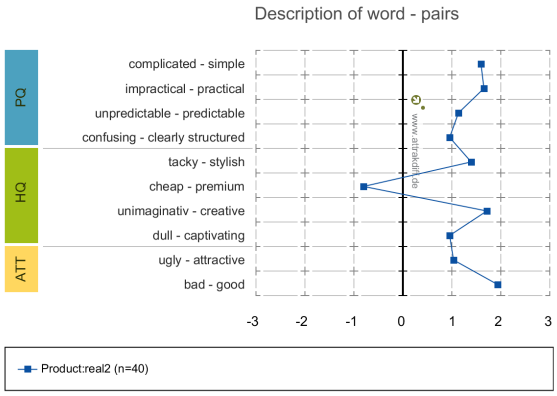


Figure 9: Description of word-pairs

model, such as a use case (Elallaoui et al., 2018), or some other natural language processing technique, with a presentation of reports that can be interpreted (e.g. AQUSA) (Lucassen et al., 2016).

Using an intermediate model to represent the User Story adds complexity to the use of the solution, which can be seen as an unwanted feature. Our approach does not use an intermediate model. Our approach makes use of text readability indexes, a technique already widely used for text analysis in other areas (e.g. economics, literature), and the use of OpenAI's LLM with personalized recommendations. In addition, the estimator module uses machine learning with natural language processing techniques.

The proposal USQA uses natural language processing techniques to analyze usefulness, completeness, and polysemes in the user stories creation (Jiménez et al., 2023). Our proposal brings additional techniques, such as recommendation and readability of the User Story that can help even more. Table 8 compares UST to the AQUSA and USQA proposals and it illustrates UST's contribution as compared to that of existing related work to User Story writing.

Table 8: Comparison with related work

Tool	**Intermediate Model**	**Recommend Report**
UST	No	Yes
AQUSA	No	No
USQA	No	Yes

7 LIMITATIONS AND THREATS

There is some criticism in the literature regarding the numerical interpretation of a Likert scale questionnaire (For example, in the calculation of the Likert scale average or mean) (Fávero and Belfiore, 2017). To minimize this point, we use another framework for analyzing software quality, the AttrakDiff.

The use of an LLM model made available by companies via API (e.g. OpenAI's ChatGPT) ties the UST solution to a corporate company. In future work, we intend to use and validate an open-source LLM model.

Readability indexes must be interpreted with caution, as their formulae use only two variables: complex words and long sentences. Therefore, they are not able to measure the cohesion and coherence of a business User Story, which covers semantic, syntactic, and pragmatic factors.

Estimation in Story Points generally follows the Fibonacci scale. In our proposal, the estimator returns a real number between 0 and 100. This problem was treated as a regression problem and not a classification one. However, we can obtain probably greater interpretability if we use the Fibonacci scale instead of real numbers.

8 CONCLUSIONS AND ONGOING WORK

This paper presented a proposal and evaluation of a tool for recommending good practices in writing User Stories with LLM, in addition to a User Story estimation module with Machine Learning and presentation of readability indexes for the User Story description. The proposed tool was evaluated by 40 software engineering practitioners. The evaluation was conducted with the TAM and AttrakDiff frameworks. Results indicate that UST meets the established objectives, with good acceptance from its intended users.

From this investigation, one may conclude that a tool to assist the construction of User Stories is a viable technique that, at the very least, can be used to educate teams on writing better User Stories. In fact, from the evaluation experiment, one may say that UST could help the User Stories by providing feedback to the agile practitioner.

The paper also presented a dataset with data from projects mined from GitLab that were used to train the predictive model for Story Points. This dataset can be used in other research related to agile software development. Work on named entity recognition to extract entities from the User Story text is ongoing. Independent future work could entail additional validation experiments including integration and evaluation of UST with computer-based education platforms for agile software development methods.

REFERENCES

Alomary, A. and Woolard, J. (2015). How is technology accepted by users? a review of technology acceptance models and theories. *IRES 17th International Conference*.

Andrade, A. F. M. (2021). *Uma Abordagem Baseada Em Gamificação Para Estimativa De Esforço Em Desenvolvimento Ágil De Software*. PhD thesis, Universidade Federal de Campina Grande.

Bobadilla, J., Ortega, F., Hernando, A., and Gutiérrez, A. (2013). Recommender systems survey. *Knowledge-based systems*, 46:109–132.

Bogert, J. (1985). In defense of the fog index. *The Bulletin of the Association for Business Communication*, 48(2):9–12.

Burke, R. (2007). Hybrid web recommender systems. *The adaptive web: methods and strategies of web personalization*, pages 377–408.

Chaparro, O., Lu, J., Zampetti, F., Moreno, L., Di Penta, M., Marcus, A., Bavota, G., and Ng, V. (2017). Detecting missing information in bug descriptions. *Proceedings of the ACM SIGSOFT Symposium on the Foundations of Software Engineering*, Part F130154:396–407.

Choetkiertikul, M., Dam, H. K., Tran, T., Ghose, A., and Grundy, J. (2018). Predicting Delivery Capability in Iterative Software Development. *IEEE Transactions on Software Engineering*, 44(6):551–573.

Choetkiertikul, M., Dam, H. K., Tran, T., Pham, T., Ghose, A., and Menzies, T. (2019). A Deep Learning Model for Estimating Story Points. *IEEE Transactions on Software Engineering*, 45(7):637–656.

Choudhury, P., Crowston, K., Dahlander, L., Minervini, M. S., and Raghuram, S. (2020). GitLab: work where you want, when you want. *Journal of Organization Design*, 9(1).

Cohn, M. (2004). *User stories applied: For agile software development*. Pearson Education.

Cohn, M. (2005). *Agile Estimating and Planning*. Pearson Education.

Dantas, E., Costa, A. A. M., Vinicius, M., Perkusich, M. B., de Almeida, H. O., and Perkusich, A. (2019). An effort estimation support tool for agile software development: An empirical evaluation. In *SEKE*, pages 82–116.

Davis, F. D., Bagozzi, R. P., and Warshaw, P. R. (1989). User acceptance of computer technology: A comparison of two theoretical models. *Management science*, 35(8):982–1003.

Dimitrijević, S., Jovanović, J., and Devedžić, V. (2015). A comparative study of software tools for user story management. *Information and Software Technology*, 57:352–368.

Dragicevic, S., Celar, S., and Turic, M. (2017). Bayesian network model for task effort estimation in agile software development. *Journal of Systems and Software*, 127:109–119.

DuBay, W. H. (2004). The principles of readability: A brief introduction to readability research. *Impact Information*, (949):1–72.

Dybå, T. and Dingsøyr, T. (2008). Empirical studies of agile software development: A systematic review. *Information and Software Technology*, 50(9-10):833–859.

Elallaoui, M., Nafil, K., and Touahni, R. (2018). Automatic transformation of user stories into uml use case diagrams using nlp techniques. *Procedia computer science*, 130:42–49.

Fávero, L. P. and Belfiore, P. (2017). *Manual de análise de dados: estatística e modelagem multivariada com Excel®, SPSS® e Stata®*. Elsevier Brasil.

Gavidia-Calderon, C., Sarro, F., Harman, M., and Barr, E. T. (2021). The Assessor's Dilemma: Improving Bug Repair via Empirical Game Theory. *IEEE Transactions on Software Engineering*, 47(10):2143–2161.

Gliem, J. A. and Gliem, R. R. (2003). Calculating, interpreting, and reporting cronbach's alpha reliability coefficient for likert-type scales. Midwest Research-to-Practice Conference in Adult, Continuing, and Community Education.

Gross, P. P. and Sadowski, K. (1985). FOGINDEX: A readability formula program for microcomputers. *Journal of Reading*, 28(7):614–618.

Hassenzahl, M., Burmester, M., and Koller, F. (2003). Attrakdiff: Ein fragebogen zur messung wahrgenommener hedonischer und pragmatischer qualität. *Mensch & Computer 2003: Interaktion in Bewegung*, pages 187–196.

Huang, Y., Wang, J., Wang, S., Liu, Z., Wang, D., and Wang, Q. (2021). Characterizing and predicting good first issues. *International Symposium on Empirical Software Engineering and Measurement*.

Jadhav, D., Kundale, J., Bhagwat, S., and Joshi, J. (2023). A Systematic Review of the Tools and Techniques in Distributed Agile Software Development. *Agile Software Development: Trends, Challenges and Applications*, pages 161–186.

Jiménez, S., Alanis, A., Beltrán, C., Juárez-Ramírez, R., Ramírez-Noriega, A., and Tona, C. (2023). Usqa: A user story quality analyzer prototype for supporting software engineering students. *Computer Applications in Engineering Education*.

Koenke, K. (1971). Another practical note on readability formulas. *Journal of Reading*, 15(3):203–208.

Lucassen, G., Dalpiaz, F., van der Werf, J. M. E., and Brinkkemper, S. (2016). Improving agile requirements: the quality user story framework and tool. *Requirements engineering*, 21:383–403.

Mani, S., Sankaran, A., and Aralikatte, R. (2019). Deeptriage: Exploring the effectiveness of deep learning for bug triaging. *ACM International Conference Proceeding Series*, pages 171–179.

Mergel, I. (2016). Agile innovation management in government: A research agenda. *Government Information Quarterly*, 33(3):516–523.

Odilinye, L. and Popowich, F. (2020). Personalized recommender system using learners' metacognitive reading activities. In *Methodologies and Intelligent Systems for Technology Enhanced Learning, 10th International Conference*, pages 195–205. Springer.

OpenAI (2023). GPT-4 Technical Report. 4:1–100.

Ortu, M., Destefanis, G., Adams, B., Murgia, A., Marchesi, M., and Tonelli, R. (2015). The JIRA repository dataset: Understanding social aspects of software development. *ACM International Conference Proceeding Series*, 2015-October.

Ortu, M., Murgia, A., Destefanis, G., Tourani, P., Tonelli, R., Marchesi, M., and Adams, B. (2016). The emotional side of software developers in JIRA. *Proceedings - 13th Working Conference on Mining Software Repositories, MSR 2016*, pages 480–483.

PMI (2017). Success Rates Rise - 2017 9th Global Project Management Survey. Technical report, PMI.

Porru, S., Murgia, A., Demeyer, S., Marchesi, M., and Tonelli, R. (2016). Estimating story points from issue reports. *ACM International Conference Proceeding Series*.

Ricci, F., Rokach, L., and Shapira, B. (2010). Introduction to recommender systems handbook. In *Recommender systems handbook*, pages 1–35. Springer.

Rigby, D. K., Sutherland, J., and Noble, A. (2018). Agile Scale: How to go from teams to hundreds. *Havard Business Review*, May-June(June):1–3.

Scott, E. and Pfahl, D. (2018). Using developers' features to estimate story points. *ACM International Conference Proceeding Series*, (106):106–110.

Soares, R. G. F. R. R. G. F. R. R. G. (2018). Effort Estimation via Text Classification And Autoencoders. In *2018 International Joint Conference on Neural Networks (IJCNN)*, volume 2018-July, pages 1–8. IEEE.

Sommerville, I. (2011). *Software engineering*. 9th edition.

StandishGroup (2015). The chaos report. http://www.standishgroup.com. Acessed 2023.

Sutherland, J. (2014). *Scrum: the Art of Doing Twice the Work in Half the Time*. Random House.

Tawosi, V., Al-Subaihin, A., Moussa, R., and Sarro, F. (2022a). *A Versatile Dataset of Agile Open Source Software Projects*, volume 1. Association for Computing Machinery.

Tawosi, V., Moussa, R., and Sarro, F. (2022b). Deep Learning for Agile Effort Estimation Have We Solved the Problem Yet? pages 1–17.

Textstat (2023). textstat/textstat: python package to calculate readability statistics of a text object - paragraphs, sentences, articles. Acessed 2023.

Torrent, T. T., Hoffmann, T., Almeida, A. L., and Turner, M. (2023). Copilots for linguists: Ai, constructions, and frames. *Elements in Construction Grammar*.

Trimble, J., Shirley, M. H., and Hobart, S. G. (2016). Agile: From software to mission system. *14th International Conference on Space Operations, 2016*, pages 1–8.

Umer, Q., Liu, H., and Illahi, I. (2020). CNN-Based Automatic Prioritization of Bug Reports. *IEEE Transactions on Reliability*, 69(4):1341–1354.

Valdez, A., Oktaba, H., Gomez, H., and Vizcaino, A. (2020). Sentiment analysis in jira software repositories. *Proceedings - 2020 8th Edition of the International Conference in Software Engineering Research and Innovation, CONISOFT 2020*, pages 254–259.

Wieringa, R. J. (2014). *Design science methodology for information systems and software engineering*. Springer.

Zach, A. (2023). Statology. https://www.statology.org/internal-consistency/. Accessed on 01.01.2023.

Apêndice F

A Predictive Model for *Story Points* leveraging features like readability and sentiment from *User Story* description

Artigo publicado na conferência ITS2025 - International Conference on Intelligent Tutoring Systems (pp. 274-284). Springer Nature Switzerland [96].

A Predictive Model for Story Points Leveraging Features Like Readability and Sentiment from User Story Description

Giseldo da Silva Neo[1,2(✉)], J. Antão B. Moura[2], Alana Viana Borgesda Silva Neo[2,3], and Evandro de Barros Costa[4]

[1] Federal Institute of Alagoas, Maceió, AL, Brazil
giseldo.neo@ifal.edu.br
[2] Federal University of Campina Grande, Campina Grande, PB, Brazil
antao@computacao.ufcg.edu.br
[3] Federal Institute of Mato Grosso do Sul, Corumbá, MS, Brazil
alana.neo@ifms.edu.br
[4] Computing Institute, Federal University of Alagoas, Maceió, AL, Brazil
evandro@ic.ufal.br

Abstract. Software development effort estimation is strategic for organizations. In Agile methodologies, Story Points (SP) are commonly used to measure the effort required to complete a user story. While machine learning techniques have been explored for effort estimation, existing models, especially those using Deep Learning, can be time-consuming for feature extraction and training. This study addresses the research gap by proposing and validating a Story Point predictor. The model uses features extracted from user story titles and descriptions, specifically focusing on readability, sentiment, and subjectivity indicators. The study used an experimental methodology with a quantitative approach, validated on a dataset of 34 open-source projects. Evaluation was performed using the Mean Absolute Error (MAE) metric and the Wilcoxon test for statistical significance. Two baselines, Mean-Based Regression (MbR) and TFIDFSE, were used for comparison. The results show that the model achieved a better (lower) apparent MAE than MbR and TFIDFSE in most of the projects. Furthermore, the proposed approach for feature extraction and training is significantly faster than the baselines. Extracting readability, sentiment, and subjectivity features from user stories shows promise in improving estimation accuracy and efficiency.

Keywords: Effort estimation · User Story · Story Points · readability

1 Introduction

Some managers believe that software estimation is an unsolvable problem [27], while others consider that estimates are reasonably accurate in some projects

S. Graf and A. Markos (Eds.): ITS 2025, LNCS 15724, pp. 274–284, 2026.
https://doi.org/10.1007/978-3-031-98284-2_21

[10]. Despite some professionals' disbelief, using estimates is strategic in organizations. For example, the Brazilian government uses Function Point (FP) as a standard for its estimates in software engineering public contracts [13]. FP is a template that uses the algorithm technique [22]. However, the need to document the features extracted from the FP model creates barriers to its adoption.

Agile software development (ASD) is a methodology that can provide flexibility in delivering products through multiple iterations of development [5]. A core component of ASD is the planning step and the estimation of the effort. Story points (SP) are the basis for the measurement in this methodology [28]. SP measures how much effort is required to complete a user story compared to previous user stories of the same project. SP is usually measured in numbers on the Fibonacci scale [23]. However, nothing prevents the team from assigning points on other scales.

Several studies have sought to solve the estimate of software development effort using machine learning techniques [1,4,18,20]. For example, some researchers concluded that the features extracted from the title and description of the user story and more information from the development team can be used to estimate the project efforts [20].

Models that extract features from the title and description of user stories with Deep Learning can take 2 to 8 h of training per project [17], so there is a research gap in predictive models that can extract features with speed. A solution is the use of some traditional machine learning methods, such as TFIDF with SVM. An example of this approach is the model that we call in this article TFIDFSE, a model used to predict SP from the title and description of the user story [18] created by Porru and other researchers. This model is was used as a baseline when new models are proposed [4].

Various features can be extracted from the title and description of the user story. One of these features is the readability indicators (for example, the Fog Index [9]). They have already been used to predict the delivery capacity of the team that develops software projects [3] and predict some characteristics of financial statements [12]. However, to our knowledge, it has not been specifically used to solve the effort estimation problem.

The objective of this study is to propose and validate a Story Point predictor model that uses user story titles and descriptions of the user story as input (aka NEOSP). This model was validated on a data set collected from GitLab top open-source projects. For predictive attributes, readability, sentiment, and subjectivity indicators extracted from the user story title and description were used. Two baselines are used for performance comparison purposes, Mean-Based Regression (MbR) and TFIDFSE [4]. The assumption is that SP can be estimated from the user story titles and descriptions if readability and sentiment indicators are used as feature extraction.

The research questions that guide this article are as follows: How well does a model with readability, sentiment, and subjectivity extracted from the title and description of tasks (aka NEOSP) perform compared to a MbR baseline?; And, how well does NEOSP perform compared to the TFIDFSE model?

2 Background

2.1 NLP Readability Indicators

The readability indicators used in the model were Gunning Fog, Flesch Reading Ease, Flesch-Kincaid Grade Leve, Coleman Liau Index, Linsear Write Formula, Dale-Chall Readability Formula, and Automated Readability Index.

Gunning Fog. The Gunning Fog Index is an indicator assigned to a given text. The higher the value, the more complex it is. The Gunning-Fog formula is presented in Eq. 1 [26].

$$0.4 \left[\left(\frac{words}{sentences} \right) + 100 \left(\frac{complex\ words}{words} \right) \right] \tag{1}$$

Flesch Reading Ease. Another text readability indicator is the FRE [26]. The higher the value, the more complex the text to read. Its maximum value is 121.22. There is no minimum value, and negative scores are also valid. Equation 2 presents the FRE calculation.

$$206,835 - 1.015 \left(\frac{total\ words}{total\ sentences} \right) - 84.6 \left(\frac{total\ syllables}{total\ words} \right) \tag{2}$$

Flesch-Kincaid Grade Level. Returns the Flesch-Kincaid Grade of the text [26]. See Eq. 3.

$$0.39 \left(\frac{total\ words}{total\ sentences} \right) + 11.8 \left(\frac{total\ syllables}{total\ words} \right) - 15.59 \tag{3}$$

Coleman Liau Index. Returns another complexity indicator using Eq. 4 [26]. L is the average number of letters per 100 words, and S is the average number of sentences per 100 words.

$$CLI = 0.0588\, L - 0.296\, S - 15.8 \tag{4}$$

Linsear Write Formula. Another indicator is the Linsear Write Formula [26]. The Lw metric needs at least 100 words of text. The algorithm works as follows: 1. For each simpler word, defined as words that are only 2 (two) syllables or less, add a dot; 2. For each more difficult word, defined as a word of 3 syllables or more, add 3 points; 3. Divide the points by the number of sentences in the 100-word sample; 4. Adjust the temporary value of r: if $r > 20$ then $Lw = r/2$; otherwise $r < 20$ then $Lw = r/2 - 1$

Dale–Chall Readability Formula. This indicator uses a lookup table of the 3000 most used English words [26]. Equation 5 shows this calculation.

$$0.1579 \left(\frac{dificult\ words}{total\ words} \times 100 \right) + 0.0496 \left(\frac{total\ words}{total\ sentences} \right) \tag{5}$$

Automated Readability Index. Returns the automated readability index from Formula 6 [26].

$$4.71 \left(\frac{characters}{words} \right) + 0.5 \left(\frac{words}{sentences} \right) - 21.43 \tag{6}$$

2.2 Subjectivity and Polarity

In addition to readability indicators, two other features were extracted from the title and description of the tasks: Subjectivity and Polarity. Sentiment analysis is a technique for identifying and categorizing opinions expressed in textual data to determine the underlying emotional tone or attitude (positive, negative, or neutral) towards a topic in a text. Subjectivity also helps in this context and refers to the linguistic expression of personal feelings, views, beliefs [16].

2.3 Evaluation Metrics

Several measures are used in the literature on the estimation of software development effort estimation to measure the precision of estimation models. These measurements generally use the predicted and actual values in their calculation [24]. The Mean Absolute Error (MAE) has already been used in other studies [11,19,21,24], so we adopt this measure as a model evaluation parameter. The MAE is calculated according to Eq. 7. E_{actual} is the current effort, $E_{predicted}$ is the predicted effort, and n is the number of observations [26].

$$MAE = \frac{1}{n} \sum_{i=1}^{n} |E_{actual} - E_{predicted}| \tag{7}$$

3 Methodology

The study employs an experimental methodology with a quantitative approach. The main steps include the following: Model Proposal and Evaluation; Data Collection and Preparation; Feature Extraction; Model Building and Training. Some of the steps of the machine learning workflow techniques suggested by Google were also used (Fig. 1).

The entire experiment was implemented in the Python language with Jupyter Notebooks and is available for reproduction on GitHub[1]. To calculate the metrics, cross-validation (k-fold with 10 partitions) was used. For comparison, the

[1] https://github.com/giseldo/neosp.

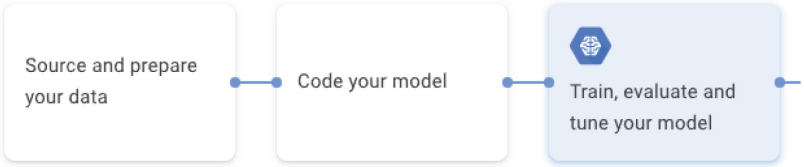


Fig. 1. Machine learning steps from Google. Source https://cloud.google.com/ai-platform/docs/ml-solutions-overview

selected metric was MAE because it was used in some related research [3,8,24]. Two baselines were used, an MbR and another technique previously presented as a solution to the same problem, the TFIDFSE [4]. The Wilcoxon test was used with the confidence interval calculation to compare the MAE of the models. The proposed model extracted features from the user story, with the support of the library "textstat" and "textblob" [25,26], and the predictive algorithm used was SVR (SVM for Regression) from Scikit-learn for the regression model.

The sentiment classification algorithm used was PatternAnalyser, available in the Pattern library, specifically in the Pattern.en module of "textblob" Python library [25]. This classification method uses an extended vocabulary from the WordNet3 dictionary, called the Duch sentiment lexicon. This dictionary has approximately 5,750 adjectives annotated with their respective sentiment polarity indicators and was trained with more than 1000 manual annotations; the remaining adjectives were annotated with the help of the unsupervised K-NN technique. The PatternAnalyser algorithm for sentiment analysis was chosen due to its speed and availability of use compared to other algorithms.

In this study a data set called NEODATASET [14,15] was used to evaluate the model. This dataset, extracted between January 2023 and April 2023, includes data from 34 projects of the top open source projects in GitLab [15]. They have different characteristics and cover various programming languages, business domains, and other geographic locations of the team. The selected projects employ agile software development methodologies and have recorded task sizes in Story Points [15]. Only user stories with the attribute "State": "closed" and that have the attribute "weight" populated were collected, because this field is used to record effort in story points. The data set was stored in CSV format and is available on GitHub[2]. An example of the data is presented in Table 1.

One model was created for each project in this data set. This implies that 34 models were created. So, the proposed model is the architecture plus data from older user stories from the same project. It was not one big model trained with data from all 34 the project. Because Story points do not make sense outside the context of that team and should not be compared between teams.

The variables originally considered in the dataset are the columns: issuekey, create, title, description, and storypoint (Table 2). IssueKey is a unique identifier created by the system when registering a Story Point. Created is the creation

[2] https://github.com/giseldo/neodataset.

Table 1. Example of a User Story from project 7764 from NeoDataset

issuekey	created	title	description	story points
29688087	17/01/20 00:50	Update templates for website merge requests	... Automatically add relevant details if possible (labels, pings, etc.) * Improve clarity of issues * Reduce grooming time * Ensure that all issues contain the [The Five W's] ...	1
29682716	16/01/20 19:21	Make sure that we Capture ... feature comparison pages	... We need it to be added to the feature comparison page, let's check if there are other items missing. > Sid: Which plan is it in? Can't find on => Add here. ## Tasks - [ ] Add elastic search to the feature comparison page ...	1

date of the User Story. The title is a short text in one sentence. The description is a more detailed text of what should be done. Story Point is a measure of effort assigned by the developer.

Table 2. Data set variables

Name	Type	Predict or Label
issuekey	categorical	removed from model
created	date	removed from model
title	text	Predict variable before extracted feature
description	text	Predict variable before extracted feature
storypoints	discrete number	Label

First, the "title" and the "description" of the user story were joined into a new column called "context". From this new column, preprocessing techniques were applied(Table 4), and then the 11 characteristics (or features) were extracted from the context. The extracted features are presented in Table 3.

The *textstat* [26] library was used to extract the readability attributes and the textblob [25] library was used to analyze the sentiment of the User Story. They are used in the text in the column contex (title + description). The language used for the experiment was Python with the sckit-learn library. The use of libraries to extract the readability indicator and sentiment with polarity is very simple and straightforward.

Table 3. Predictive Attributes extracted from user story text (title and description)

ID	Feature Name	Library and Method used
1	gunning_fog	textstat.gunning_fog
2	flesch_reading_ease	textstat.flesch_reading_ease
3	flesch_kincaid_grade	textstat.flesch_kincaid_grade
4	smog_index	textstat.smog_index
5	coleman_liau_index	textstat.coleman_liau_index
6	automated_readability_index	textstat.automated_readability_index
7	dale_chall_readability_score	textstat.dale_chall_readability_score
8	difficult_words	textstat.difficult_words
9	linsear_write_formula	textstat.linsear_write_formula
10	polarity	textblob.sentiment.polarity
11	subjectivity	textblob.sentiment.subjectivity

Table 4. Preprocessing technique used in the user story text (title and description)

Name of technique	Regular Expression
remove stopwords	-
remove extra whitespace tabs	`\s+`
remove punctuation	`[\w\s]`
remove numbers	`\b\w*\d\w*\b`
remove special characters	`[^A-Za-z0-9\s]`
remove html tags	`<[^>]+>`
remove URLs content	`http\S+|www\S+`

4 Results and Discussion

NEOSP vs. MbR: The NEOSP model achieved an MAE (apparently) better than an MbR regressor (a dummy mean regressor) in almost all designs. However, in 1 out of 34 projects, MbR performed better than NEOSP. More research is needed to understand this point, analyzing this project individually. NEOSP had a better (or lower) MAE than MbR in 33 projects out of 34. However, having a better MAE does not necessarily mean that one model is better or worse than another, as this is an apparent measure. So, the Wilcoxon test was used to obtain the result of the 10 MAE returned from cross-validation. For 27 projects, the Wilcoxon test had a p-value lower than 0.1 when comparing the mean MAE of the NEOSP model with the MAE of the MbR model. In these 27 projects, we reject the null hypothesis that there is no difference between means. The alternative hypothesis of the test is that there is a difference between the means. The null hypothesis was rejected, so we accept the alternative hypothesis.

Therefore, for all these 27 projects, the NEOSP MAE was better than the MbR MAE with statistical significance.

NEOSP vs. TFIDFSE: The NEOSP model achieved an MAE (apparently) better than the TFIDFSE in almost all designs. In the 34 projects, 29 had a better MAE for NEOSP and 5 had a better MAE for TFIDFSE. Having a better average MAE does not necessarily mean that a model is different, better, or worse, as this is an apparent measure. So, the Wilcoxon test was used. For 12 projects of 34, the Wilcoxon test had a p-value lower than 0.1 when comparing the MAE average of the NEOSP model with the TFIDFSE model. So in these 12 (eight) projects, we reject the null hypothesis that there is no difference between the means.

Implications: There is no significant difference between the TFIDFSE approaches [18] and Deep Learning (DeepSE) [4], as reproduced by other researchers [24], who compared the two approaches with other data (TAWOS Dataset). Therefore, since NEOSP had results close to TFIDFSE, similar results would be found if the technique was reproduced and compared with DeepSE, but this experiment must be performed to confirm this hypothesis.

Rutime: In a couple of seconds, all features were extracted from the title and description of the tasks of all projects. One project with the most tasks took more than half the total time. The proposed approach is much faster than the baseline compared. The 34 models were created, and the metrics were extracted in a couple of seconds.

5 Related Works

Some studies with proposals to predict the estimation of effort in agile projects are presented in this section.

[1]: This study created an SP predictor based on the description of the user story. Regression Models, Neural Networks [2], and Support Vector Machine for Regression (SVR) were evaluated using cross-validation [6] and metrics such as RMSE, MRE, and PRED(x) [7]. The study used two case studies with 1,325 and 13 user stories, respectively. The features extracted from the user story description included character count, frequency of the 15 most common words, and priority (total of 17 features). The results were better in the smaller case study, which had more structured texts, leading to the conclusion that the structure of the text improves prediction. SVR and Linear Regression achieved the best RMSE results in this case.

[18]: This work developed an SP predictor based on the user story description, comparing algorithms such as SVM, KNN, Decision Tree, and Naive Bayes. They used accuracy and MMRE [28] as metrics, with the ZeroR algorithm as a baseline. SVM showed the best MMRE results. The sample consisted of 5,607 user stories from JIRA projects (after filtering from 16,523). The title and description were combined ("Context") and the feature extraction was performed using TFIDF after the code was separated from the text. The approach was classification (SP as classes), not regression. They concluded that the user story

description contains relevant information and that training with more than 300 user stories leads to a better MMRE (>0.61).

[20]: This study analyzed features extracted from the developer (reputation, workload, total capacity) to predict SP. They compared models using only developer features, only text features, both, and baselines such as Random Guessing, Mean, and Median. They used Accuracy, MAE, and Standard Accuracy for evaluation. The sample was 4,142 tasks (after cleaning from 15,155), using the same data set as [18]. They also used a classification strategy with SVM, similar to [18]. They concluded that the model with developer features outperforms the model based solely on text features.

6 Threats to Validity

The data set used may not represent all characteristics of the project. By definition, a project is unique, but may share common characteristics; using projects with diverse characteristics is necessary to confirm the predictive results of the proposed model.

There are indications that the task of estimating story points is a task that depends on more information than is available in the task description, according to [20]. More studies are needed to confirm these findings that isolate other variables.

The experimental evaluation lacks comparisons with more complex and standard architectures. A fair comparison with modern LLM (both considering computational costs and achieved quality) would be highly beneficial. Finally, one can argue that building a more accurate model, which could be applied for years resulting in improving the efficiency of software engineering in a company, is definitely worth a higher training time in a trade-off for accuracy.

7 Final Considerations

We proposed a Story Points predictive model that extracts readability features from task titles and descriptions. The model was trained with open source projects extracted from GitLab. For some projects, NEOSP performed better with project estimation than MbR regression and TFIDFSE.

The proposed model demonstrates significant improvements over the baselines, particularly in terms of MAE. The use of features such as readability, sentiment, and subjectivity in user story descriptions is interesting and shows promise for improving the accuracy of the estimation. In addition, NEOSP requires less time to extract readability features and train the model, which is much more efficient than the compared baselines.

For future research, we suggest new techniques to improve model performance through other text representation and normalization strategies and comparison with other baselines, like LSTM-based approaches, pre-trained Transformer-based architectures such as BERT and LLM GPTs. Further research could explore the applicability of NEOSP in different development environments and conditions in other data sets.

Artifact Availability

The source code for reproduction of NEOSP is available at https://github.com/giseldo/neosp. The data set NEODATASET at https://github.com/giseldo/neodataset

Acknowledgements. This work was carried out with the support of the Coordination for the Improvement of Higher Education Personnel - Brazil (CAPES) - Financing Code 001.

References

1. Abrahamsson, P., Fronza, I., Moser, R., Vlasenko, J., Pedrycz, W.: Predicting development effort from user stories. In: 2011 International Symposium on Empirical Software Engineering and Measurement. pp. 400–403 (2011). https://doi.org/10.1109/ESEM.2011.58
2. Bishop, C.M.: Neural Network for Pattern Recognition (1995)
3. Choetkiertikul, M., Dam, H.K., Tran, T., Ghose, A., Grundy, J.: Predicting delivery capability in iterative software development. IEEE Trans. Softw. Eng. **44**(6), 551–573 (2018). https://doi.org/10.1109/TSE.2017.2693989
4. Choetkiertikul, M., Dam, H.K., Tran, T., Pham, T., Ghose, A., Menzies, T.: A deep learning model for estimating story points. IEEE Trans. Softw. Eng. **45**(7), 637–656 (2019). https://doi.org/10.1109/TSE.2018.2792473
5. Cohn, M.: Agile Estimating and Planning (2005)
6. Faceli, K., Lorena, A.C., Gama, J., de Carvalho, A.C.: Inteligência artificial: uma abordagem de aprendizado de máquina, 1st edn. LTC - Livros Técnicos e Científicos Editora LTDA, Rio de Janeiro (2011)
7. Foss, T., Stensrud, E., Kitchenham, B., Myrtveit, I.: A simulation study of the model evaluation criterion MMRE. IEEE Trans. Softw. Eng. **29**(11), 985–995 (2003). https://doi.org/10.1109/TSE.2003.1245300
8. Fu, M., Tantithamthavorn, C.: GPT2SP: a transformer-based agile story point estimation approach. IEEE Trans. Softw. Eng. **49**(2), 611–625 (2023). https://doi.org/10.1109/TSE.2022.3158252
9. Gross, P.P., Sadowski, K.: FOGINDEX: a readability formula program for microcomputers. J. Read. **28**(7), 614–618 (1985)
10. Kitchenham, B., Lawrence Pfleeger, S., McColl, B., Eagan, S.: An empirical study of maintenance and development estimation accuracy. J. Syst. Softw. **64**(1), 57–77 (2002). https://doi.org/10.1016/S0164-1212(02)00021-3
11. Langdon, W.B., Dolado, J., Sarro, F., Harman, M.: Exact mean absolute error of baseline predictor, MARP0. Inf. Softw. Technol. **73**, 16–18 (2016). https://doi.org/10.1016/j.infsof.2016.01.003
12. Lo, K., Ramos, F., Rogo, R.: Earnings management and annual report readability. J. Account. Econ. **63**(1), 1–25 (2017). https://doi.org/10.1016/j.jacceco.2016.09.002
13. Ministério do Planejamento, D.e.G.: Roteiro Métricas de Software do SISP (2018)
14. Neo, G.S., Moura, J., Almeida, H., Neo, A., Freitas Júnior, O.: User story tutor (UST) to support agile software developers. In: Proceedings of the 16th International Conference on Computer Supported Education - Volume 2: CSEDU, pp. 51–62. INSTICC, SciTePress (2024). https://doi.org/10.5220/0012619200003693

15. Neo, G.S., Neo, A.V.B.d.S., Filho, K.J.A.G., Moura, J.A.B., Junior, O.d.G.F.: NeoDataset: um conjunto de dados com user stories e story points. Revista dos Mestrados Profissionais **133**(2), 194–211 (2024). https://doi.org/10.51359/2317-0115.2024.265431
16. Pang, B., Lee, L., et al.: Opinion mining and sentiment analysis. Found. Trends Inf. Ret. **2**(1–2), 1–135 (2008)
17. Phan, H., Jannesari, A.: Heterogeneous graph neural networks for software effort estimation. In: Proceedings of the 16th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement, pp. 103–113. ESEM 2022. Association for Computing Machinery, New York (2022). https://doi.org/10.1145/3544902.3546248
18. Porru, S., Murgia, A., Demeyer, S., Marchesi, M., Tonelli, R.: Estimating story points from issue reports. In: Proceedings of the The 12th International Conference on Predictive Models and Data Analytics in Software Engineering. PROMISE 2016. Association for Computing Machinery, New York, NY, USA (2016). https://doi.org/10.1145/2972958.2972959
19. Sarro, F., Petrozziello, A., Harman, M.: Multi-objective software effort estimation. In: Proceedings of the 38th International Conference on Software Engineering, ICSE 2016, pp. 619–630. Association for Computing Machinery, New York, NY, USA (2016). https://doi.org/10.1145/2884781.2884830
20. Scott, E., Pfahl, D.: Using developers' features to estimate story points. In: Proceedings of the 2018 International Conference on Software and System Process, ICSSP 2018, pp. 106–110. Association for Computing Machinery, New York, NY, USA (2018). https://doi.org/10.1145/3202710.3203160
21. Shepperd, M., MacDonell, S.: Evaluating prediction systems in software project estimation. Inf. Softw. Technol. **54**(8), 820–827 (2012). https://doi.org/10.1016/j.infsof.2011.12.008
22. Sudarmaningtyas, P., Mohamed, R.: A review article on software effort estimation in agile methodology. Pertanika J. Sci. Technol. **29**(2), 837–861 (2021). https://doi.org/10.47836/pjst.29.2.08
23. Tamrakar, R., Jørgensen, M.: Does the use of Fibonacci numbers in planning poker affect effort estimates? In: 16th International Conference on Evaluation & Assessment in Software Engineering, EASE 2012, pp. 228–232 (2012). https://doi.org/10.1049/ic.2012.0030
24. Tawosi, V., Moussa, R., Sarro, F.: Agile effort estimation: have we solved the problem yet? Insights from a replication study. IEEE Trans. Softw. Eng. **49**(4), 2677–2697 (2023). https://doi.org/10.1109/TSE.2022.3228739
25. TextBlob: TextBlob: Simplified Text Processing (2024). https://textblob.readthedocs.io/en/dev/
26. Textstat: textstat/textstat: Python package to calculate readability statistics of a text object - paragraphs, sentences, articles (2023). https://github.com/textstat/textstat
27. Uc-Cetina, V.: Recent Advances in Software Effort Estimation using Machine Learning, pp. 1–10 (2023). http://arxiv.org/abs/2303.03482
28. Usman, M., Mendes, E., Weidt, F., Britto, R.: Effort estimation in agile software development: a systematic literature review. In: Proceedings of the 10th International Conference on Predictive Models in Software Engineering, PROMISE 2014, pp. 82–91. Association for Computing Machinery, New York, NY, USA (2014). https://doi.org/10.1145/2639490.2639503

Apêndice G

Estimativa de Esforço em *Story Points* a partir de *User Stories* com Large Language Models

Artigo publicado na conferência CBSOFT 2025, Congresso Brasileiro de Software (CBSoft), na trilha “SBES 2025 - XXXIX Simpósio Brasileiro de Engenharia de Software” [97].

Estimativa de Esforço em Story Points a partir de User Stories com Large Language Models

Giseldo da Silva Neo
Instituto Federal de Alagoas
Viçosa, Alagoas, Brasil
giseldo.neo@ifal.edu.br

José Antão Beltrão Moura
Universidade Federal de Campina Grande
Campina Grande, Paraíba, Brasil
antao@computacao.ufcg.edu.br

Alana Viana Borges da Silva Neo
Instituto Federal do Mato Grosso do Sul
Corumba, Mato Grosso do Sul, Brasil
alana.neo@ifms.edu.br

Olival de Gusmão Freitas Júnior
Universidade Federal de Alaagoas
Maceió, Alagoas, Brasil
olival@ic.ufal.br

RESUMO

A estimativa de esforço em projetos ágeis continua sendo um desafio recorrente, especialmente quando os story points precisam ser inferidos apenas a partir do texto das user stories. Estudos anteriores focaram principalmente em abordagens de aprendizagem de máquina para predizer o esforço, mas a recente disponibilidade de Large Language Models (LLMs) oferece uma alternativa. O objetivo do artigo é investigar a eficácia dos LLMs em estimar story points. Um derivado do modelo BERT foi ajustado (fine-tuning) e comparado, em relação ao erro médio absoluto, a três baselines: (i) um modelo preditivo tradicional baseado em vetores TF-IDF acoplados a um classificador de Regressão Linear, (ii) um modelo LLM Zero Shot, e (iii) um modelo LLM few shot. Foi utilizado um conjunto de dados de user stories de projetos reais de desenvolvimento de software ágil, o Deep-SE, um dataset com várias User Stories de 16 projetos open-source diferentes retirados do Jira. Os resultados mostram que o LLM ajustado teve MAE menor na maioria dos projetos. Os achados sugerem que, apesar do custo computacional maior, LLMs constituem uma alternativa com menor erro para a estimativa de esforço do que as técnicas comparadas.

PALAVRAS-CHAVE

Estimativa de esforço, Story points, User story, Large language model

1 Introdução

No desenvolvimento ágil de software, a estimativa de esforço é um processo necessário para o planejamento e gerenciamento de projetos [5]. Equipes ágeis frequentemente utilizam story points para representar de forma abstrata o tamanho ou complexidade de user stories e tarefas no backlog do produto [7]. Os story points são tipicamente atribuídos através de consenso humano em sessões de planning poker, baseadas na experiência e julgamento [23].

Embora essa abordagem colaborativa seja amplamente adotada, ela está sujeita a vieses humanos e variação entre equipes [7]. Estimativas com alta margem de erro podem levar a atrasos em cronogramas e estouro de custos, problemas ainda comuns mesmo em projetos ágeis [29]. Assim, há um crescente interesse em automatizar ou apoiar a estimativa de story points por meio de técnicas de aprendizado de máquina e processamento de linguagem natural, aproveitando a riqueza de dados textuais presentes nas descrições das user stories [6, 13, 29].

Nos últimos anos, modelos de linguagem de larga escala (Large Language Models, LLMs) surgiram como ferramentas promissoras para lidar com tarefas de processamento de linguagem natural complexas [15]. Modelos pré-treinados como BERT e GPT demonstraram capacidade de compreender nuances semânticas em texto e transferir esse conhecimento para diversas tarefas específicas através de fine-tuning [9].

Diante desse progresso, pesquisadores passaram a investigar se LLMs poderiam melhorar a predição de story points a partir da descrição textual das user stories, em comparação com métodos tradicionais de estimativa de esforço [2, 26].

Este artigo tem como objetivo avaliar modelos LLM (zero-shot, few-shot e fine-tuned) para estimar esforço em Story Points. Contrastando seu desempenho preditivo com a abordagem clássica regressão linear com extração de features com a técnica frequency inverse document frequency (TF-IDF). A hipótese é que um modelo de LLM ajustado (fine-tuning) traga um erro médio absoluto (MAE) menor do que os baselines selecionados.

O modelo utilizado foi o distilbert-base-uncased, este oferece um bom equilíbrio entre qualidade semântica, eficiência computacional e facilidade de ajuste para um problema de regressão sobre textos, permitindo treinar e implantar um modelo de estimativa de esforço mesmo em hardware modesto e com datasets medianos.

Nosso modelo ajustado, o distilbert-base-uncased-story-point, não perdeu poder de generalização quando treinado com dados cross-project. Esta generalização traduziu-se em resultados concretos: o modelo ajustado superou o LLM few-shot em 14 de 16 projetos e o baseline TF-IDF + RL em 11 dos 16 projetos na métrica MAE.

2 Fundamentação Teórica

Abordagens para estimativa: Antes do advento dos LLMs e do aprendizado profundo no contexto de estimativas ágeis, diversas abordagens tradicionais de aprendizado de máquina foram exploradas para prever story points a partir de dados históricos de projetos [3, 14, 20]. Essas abordagens costumam tratar a estimativa como um problema de regressão supervisionada (quando os story points são valores contínuos ou ordinais) ou de classificação em faixas (ou discretização) de esforço.

Já foi proposto para o problema de estimativa de story point a partir do texto da user story o uso de vetores de características de texto baseados em TF-IDF das descrições de issues do Jira, combinados com um classificador SVM para prever o número de story [14]. Nesse trabalho, os autores relataram naquele momento (2016) resultados promissores, superando estimativas baselines simples como usar a média ou mediana. Essa técnica serviu de referência inicial, contra a qual trabalhos subsequentes com deep learning seriam comparados [29].

Outra proposta exploraram algoritmos de regressão, ensemble e máquinas de vetores de Suporte [22]. Por exemplo, alguns pesquisadores aplicaram máquinas de vetores de Suporte usando atributos textuais e atributos de legibilidade [10]. Um outro modelo investigou até que ponto características dos desenvolvedores poderiam explicar a variância nos story points [20]. De modo geral, esses métodos mais tradicionais, obtiveram desempenho limitado, muitas vezes com erros absolutos médios altos, indicando dificuldade em capturar a complexidade semântica das user stories de usuário apenas com features lineares ou de contagem de palavras [6].

Zero Shot: Um modelo LLM em resumo é um grande modelo treinado com rede neural [16], geralmente com a tecnologia transformers [27]. A abordagem LLM zero-shot learning refere-se à capacidade dos LLMs de resolver tarefas sem a necessidade de exemplos explícitos fornecidos durante a inferência. Nesse contexto, a tarefa de processamento de linguagem natural, text classification é especificada unicamente por meio de uma instrução textual (prompt), e o modelo deve inferir a ação esperada com base em seu conhecimento prévio adquirido durante o pré-treinamento [15].

Few Shot: Já o few-shot learning caracteriza-se pela inclusão de um pequeno conjunto de exemplos da tarefa no próprio prompt, com o objetivo de guiar a geração do modelo durante a inferência. Essa técnica permite ao modelo identificar padrões desejados com base nos exemplos fornecidos e aplicá-los a novos casos, mesmo sem reconfiguração ou ajuste de parâmetros. Trata-se de uma abordagem intermediária entre o zero-shot e o treinamento supervisionado tradicional, sendo especialmente eficaz em tarefas de classificação com variações contextuais [16]. Sua principal vantagem está na adaptação rápida a novas tarefas com custo computacional reduzido.

3 Metodologia

A metodologia do estudo foi experimental e aplicada [28]. Usamos a técnica de predição direta. Os modelos preditivos construídos preveem um story point contínuo que é arredondado para o número da sequência de Fibonacci [24] mais próximo. Os procedimentos realizados em alto nível são apresentados na Figura 1.

Conjunto de dados: Conduzimos um experimento com o conjunto de dados Deep-SE [4]. O conjunto de dados utilizado tem 23.313 user stories de 16 projetos diferentes com 5 colunas. Consolidamos o dataset em um arquivo CSV e o disponibilizamos no Hugging Face (link na seção de disponibilidade dos artefatos) para facilitar o acesso e a reprodutibilidade dos experimentos. Os dados incluem o nome do projeto, o id identificador (chave), a descrição da user story, o título da user story e o story point. Um exemplo das primeiras colunas do conjunto de dados é apresentado na Tabela 1.

O conjunto de dados utilizado foi consolidado por Choetkiertikul et al. [4]. Eles disponibilizaram um link https://github.com/SEAnalytics/DSL_reading_list para o dataset, mas, no momento do acesso ao link, em maio de 2025, o dataset não estava disponível para download. Porém Tawosi et al. [25], reproduziram o experimento de Choetkiertikul com o conjunto de dados Deep-SE e re-disponibilizaram este conjunto de dados no link https://figshare.com/s/709c7e18c52e4264b70e. Estes foram os arquivos que foram utilizados, consolidados e disponibilizados no Hugging Face.

Estes dados do Deep-SE foram coletados de 16 projetos de código aberto extraídos do Jira [4] . Quanto ao domínio e às empresas, os 16 projetos são provenientes de 9 diferentes repositórios ou organizações de código aberto. Os projetos do dataset não são da mesma empresa, mas sim de diversas entidades. Os projetos demonstram diversidade em termos de domínios de aplicação e características técnicas, incluindo diferentes linguagens de programação. Todas as user stories foram criadas e registradas no sistema de rastreamento de problemas JIRA. O JIRA é um dos sistemas amplamente utilizados que suportam o desenvolvimento ágil, incluindo a estimativa de pontos de história.

Técnicas: Foram utilizadas as técnicas zero-shot, few-shot e Fine-tuning. Para o Fine-tuning, foi criado um único modelo cross-project. Para os outros casos, foram criados 16 modelos, um para cada projeto do dataset. O Fine-tuning do modelo LLM foi realizado com o modelo BERT, especificamente o distilbert-base-uncased [17]. Foi utilizado o Google Colab fornecido pelo Google para o Fine-tuning do modelo. O Fine-tuning do modelo durou poucos minutos em um processador A100. Para a extração das métricas com o modelo zero-shot e few-shot, foi utilizado o modelo Gemma3:4b em um PC comum com ollama, Visual Studio Code e Python.

Foi escolhido o modelo Gemma3:4b por sua natureza *open weight*, leveza computacional e desempenho competitivo em tarefas de compreensão de linguagem natural. Esse modelo foi projetado para rodar localmente com baixo consumo de memória, sendo compatível com ambientes de execução otimizados, permitindo a execução eficiente mesmo em computadores pessoais. Essa característica é essencial para viabilizar experimentos com LLMs sem dependência de GPUs de alto desempenho ou serviços em nuvem. Além disso, o Gemma3:4b oferece suporte nativo para abordagens few-shot e zero-shot, permitindo a realização de inferências com base em poucos exemplos ou apenas instruções em linguagem natural, sem necessidade de reconfiguração dos pesos do modelo [15, 16]. A inclusão desse modelo no experimento visa explorar alternativas acessíveis e reprodutíveis para a estimativa de esforço baseada em *user stories*, complementando a avaliação de modelos ajustados via *fine-tuning* como o *distilbert-base-uncased* [17]. Dessa forma, é possível comparar diferentes paradigmas de aplicação de LLMs no contexto de estimativas ágeis com base textual, ampliando o entendimento sobre custo-benefício e desempenho em diferentes cenários computacionais.

Pré-processamento: Para o pré-processamento, foram utilizadas as técnicas de remoção das tags HTML e a remoção das URLs, além da tokenização.

Treino e teste: Em todos os casos, a separação do conjunto de dados e do conjunto de treino foi de 70% para treino e 30% para testes (geração das métricas). Foram utilizados os primeiros 70% do conjunto de dados estratificados por projetos (nota: o conjunto de dados estava ordenado por projeto e por data de inclusão da

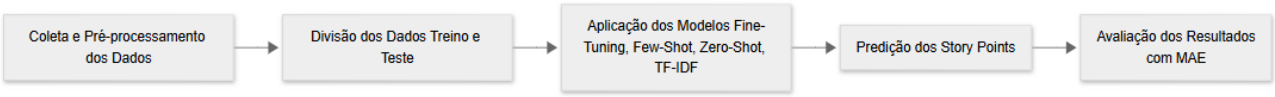


Figura 1: Procedimentos realizados

Projeto	Chave	Título	Descrição	Story Point
appceleratorstudio	TISTUD-6	Add CA against object literals in function...	The idea here is that if our met...	1
appceleratorstudio	TISTUD-9	Update branding for Appcelerator plugin...	At least fix feature icons, asso...	1
appceleratorstudio	TISTUD-11	Create new JSON schema for SDK team	Create JSON schema containing pr...	1
appceleratorstudio	TISTUD-13	Create Project References Property Page	Create property page for project...	1
appceleratorstudio	TISTUD-16	New Desktop Project Wizard	Desktop (need to convert existin...	1

Tabela 1: Amostra do conjunto de dados

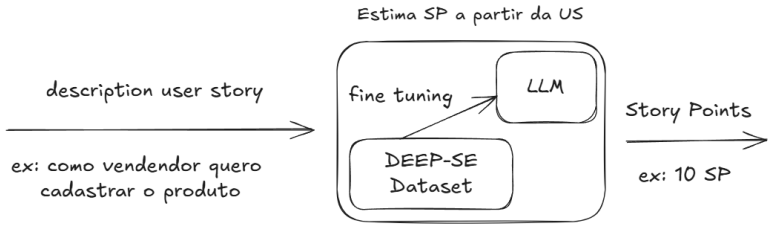


Figura 2: Arquitetura alto nível da proposta

user story, ou seja, os 70% primeiros registros do projeto estão ordenados por ordem temporal - da mais antiga para a mais recente em cada projeto). O modelo ajustado (para fine-tuning e para todos os outros) só conhece os 70%. Ou seja, os 30% utilizados para geração das métricas não eram conhecidos por nenhum dos modelos criados.

Métricas: Foi usado o erro médio absoluto (Mean Absolute Error, MAE) como métrica de comparação entre os modelos. O MAE já foi utilizado em outros estudos [8, 19, 21, 25], portanto, adotamos essa medida como parâmetro de avaliação dos modelos. O MAE é calculado conforme a Equação 1. Onde o E_{atual} é o esforço atual, e o $E_{predito}$ é o esforço predito, e n é o número de observações. Onde i é a enésima observação.

$$MAE = \frac{1}{n}\sum_{i=1}^{n}\left|E_{atual_i} - E_{predito_i}\right| \quad (1)$$

Uma visão de alto nível da previsão é apresentada na Figura 2.

Prompt: Os prompts utilizados foram construídos com base em boas práticas, tais como: contextualização da tarefa, uso de exemplos explícitos, e reforço da instrução por redundância. Foram realizados testes preliminares empíricos com variações da redação do prompt, até se chegar a uma versão que maximizasse a clareza das instruções e a consistência das respostas.

Distilbert-base-uncased

O modelo distilbert-base-uncased [1] foi a escolha para estimar esforço a partir de User Story por diversas razões. Ele foi lançado em 2019, sendo menor e mais rápido que o BERT e otimizado para tarefas que processam frases ou sentenças. Sua versão uncased é útil para descrições de código e problemas, já que não considera diferenças de maiúsculas/minúsculas. Ele é pré-treinado em um grande corpus, o que ajuda na generalização, e tem uma boa arquitetura para esse tipo de tarefa, além de exigir hardware acessível.

O distilbert resulta de um processo de knowledge distillation que reduz em aproximadamente 40% o número de parâmetros e acelera a inferência em cerca de 60%, preservando 95–97% da acurácia do BERT-base (original) em benchmarks de compreensão de linguagem natural [17]. Essa compacidade de fornecer representações ricas com custo computacional inferior é particularmente vantajosa em ambientes de hardware moderado. Com apenas 67M de parâmetros [17], a variante uncased cabe em uma GPU modesta (por exemplo, 6 GB de RAM da placa de vídeo), possibilitando fine-tuning e inferência dentro de recursos computacionais restritos. Assim, torna-se viável re-treinar o modelo à medida que novos dados de projeto se acumulam, mantendo a acurácia sem investir em infraestruturas onerosas.

Do ponto de vista linguístico, as user stories analisadas possuem extensão média inferior a 225 tokens (quando usado o tokenizador tiktoken com GPT-2); especificamente 69% das User Stories estão abaixo de 128 tokens (7215 estão acima de 128 tokens e 16098 abaixo). Logo, a maioria das User Stories é observada pelo transformer sem truncamento (abaixo de 128 tokens), preservando dependências de longo alcance.

Por fim, o amplo suporte no ecossistema Hugging Face para os modelos do tipo BERT e para outros modelos simplifica a reprodutibilidade e a integração em pipelines de tarefas de processamento de linguagem natural. Pensando nisso, o modelo gerado (distilbert-base-uncased-story-point) foi disponibilizado no Hugging Face para uso por outros pesquisadores,link disponível na seção disponibilidade dos artefatos.

Pipeline Fine Tuning

O pipeline de treinamento utilizando o modelo distilbert-base-uncased teve o objetivo de prever o valor de story points a partir da descrição textual de tarefas em projetos ágeis. O processo envolveu a preparação dos dados, tokenização, definição do modelo, configuração de treinamento e avaliação de desempenho.

Primeiramente, o conjunto de dados Deep-SE foi dividido em 70% para o subconjunto de treino e teste inicial. Na etapa de tokenização, foi aplicada a tokenização BERT com truncamento e padding para limitar as sequências a 128 tokens. Para a construção do modelo, utilizou-se um modelo distilbert com uma única saída contínua para prever valores reais (regressão), que depois foram arredondados para a sequência de Fibonacci.

Na configuração do Treinamento, definiu-se o uso de otimização com taxa de aprendizado, tamanho de batch, número de épocas e estratégias de salvamento e avaliação, respectivamente, 2e-5, 8, 4 e epoch.

A métrica usada para avaliação interna foi o erro quadrático médio (MSE) - neste caso 80% 20%. Note que a métrica MSE é utilizada internamente na construção do modelo de fine-tuning. A métrica utilizada para comparar com os outros baselines foi o erro médio absoluto, dada a facilidade de interpretação do erro médio absoluto em relação ao erro médio quadrático e seu uso na comparação entre modelos neste domínio por outros pesquisadores.

Neste pipeline foi gerado um único modelo cross-project que foi utilizado para a extração das métricas, diferente dos outros pipelines que foram criados 16 preditores, um para cada projeto. Neste pipeline, o conjunto de treinamento foi uma amostra estratificada por projeto.

Pipeline Few Shot

Para um dos modelos do baseline, foi implementada uma abordagem LLM com few-shot learning. Foi utilizado o modelo Gemma3:4b com o suporte do ollama (software para execução de LLMs em hardware local). Essa abordagem consiste em construir um modelo capaz de prever o número de story points a partir da descrição textual das user stories, fornecendo como contexto exemplos representativos de estimativas anteriores. Foram criados 16 preditores (ou modelos preditivos) e avaliados (os 16 preditores) individualmente com o MAE. Inicialmente, foi realizado o pré-processamento do conjunto de dados. Foram removidas aquelas user stories com valores nulos ou iguais a zero. O conjunto de dados resultante foi particionado em dois subconjuntos: 70% para treinamento e 30% para teste.

Para configurar o cenário de few-shot learning, foi selecionada uma amostra de 10% do conjunto de treinamento, a qual serviu como base para a construção de exemplos de entrada. Cada exemplo incluía o texto da user story, sua descrição e o valor real de story points atribuídos. Esses exemplos foram utilizados como contexto em prompts estruturados que instruem o modelo a realizar uma nova estimativa. O prompt utilizado é apresentado na Lista 1. Note que na Lista 1, no final do prompt, há uma redundância para reforçar que o modelo siga a instrução, o que é uma técnica bastante utilizada na criação de prompts: repetir o texto para enfatizar a importância da instrução.

Lista 1: Prompt Few Shot

```
Você é um especialista em estimativa de esforço
para projetos ágeis. Com base no texto da user
story e sua descrição, estime os story points
necessários para completar a tarefa. Considere
fatores como complexidade, volume de trabalho e
riscos implícitos. Retorne apenas um número
inteiro representando os story points (ex.: 1, 2,
3, 5, 8, 13, etc.). Aqui estão alguns exemplos de
estimativas anteriores:
<AQUI LOOP COM 10% DAS ESTIMATIVAS>
    Exemplo:
    Texto: {'user_story_text'}
    Descrição: {'description'}
    Story Points: {'real_story_points'}
<FIM LOOP>
Agora, estime os story points para:
Texto da user story: {user_story_data['
user_story_text']}. Descrição da user story: {
user_story_data['description']}. Retorne sempre
apenas um número inteiro representando os story
points (ex.: 1, 2, 3, 5, 8, 13, etc.). Não retorne
 nenhum texto além do número inteiro. Não retorne
nenhum texto além do número inteiro.
```

Em seguida, as tarefas do conjunto de teste foram processadas individualmente. Para cada uma delas, foi construída uma requisição textual ao modelo, incluindo os exemplos anteriores, o título e a descrição da nova tarefa, solicitando como resposta apenas um número inteiro correspondente à estimativa de esforço. O modelo LLM, configurado com baixa variabilidade (baixa temperatura/aleatoriedade: 0,3), retornava uma predição para cada tarefa, que era, então, registrada para posterior análise.

Os resultados foram organizados em uma estrutura tabular contendo identificadores das tarefas, valores reais e valores estimados, viabilizando a análise quantitativa do desempenho preditivo da abordagem. Essa estratégia foi aplicada em múltiplos projetos distintos, com o objetivo de avaliar a robustez e a capacidade de generalização do modelo em diferentes domínios de desenvolvimento ágil de software.

Pipeline Zero-shot

O pipeline zero-shot é bem semelhante ao few-shot, também foi utilizado o modelo Gemma3:4b com o suporte do ollama, porém, sem passar dados auxiliares para o LLM realizar a estimativa. Na Lista 2 é apresentado o prompt utilizado neste baseline.

Lista 2: Prompt Zero Shot

```
Você é um especialista em estimativa de esforço
para projetos ágeis. Com base no texto da user
story e sua descrição, estime os story points
necessários para completar a tarefa. Considere
fatores como complexidade, volume de trabalho e
riscos implícitos. Retorne apenas um número
inteiro representando os story points (ex.: 1, 2,
3, 5, 8, 13, etc.). Texto da user story: {'
user_story_text'}, Descrição da user story: {'
description'}. Retorne sempre apenas um número
inteiro representando os story points (ex.: 1, 2,
3, 5, 8, 13, etc.). Não retorne nenhum texto além
do número inteiro. Não retorne nenhum texto além
do número inteiro.
```

Pipeline TF-IDF-RL

Neste pipeline foi implementada uma combinação de Term Frequency–Inverse Document Frequency (TF-IDF) com Regressão Linear. O objetivo foi o mesmo dos outros pipelines, predizer os story points atribuídos a user stories a partir do texto presente nos campos de título e descrição da user story. Inicialmente, os dados foram extraídos do dataset Deep-SE. Para cada projeto selecionado, as instâncias com valores nulos ou story points iguais a zero foram descartadas. A seguir, foi criada uma representação textual unificada denominada contexto, e o mesmo procedimento foi realizado para os outros pipelines, resultado da concatenação do título e da descrição de cada user story em um único atributo textual. Os dados foram divididos em dois subconjuntos: 70% para treinamento e 30% para teste. A vetorização textual foi realizada com o método TF-IDF, que transforma os textos em uma matriz numérica ponderando a importância dos termos no corpus. A fase de modelagem envolveu o ajuste de um modelo de Regressão Linear aos dados vetorizados do conjunto de treinamento. As predições geradas sobre o conjunto de teste foram arredondadas para o inteiro mais próximo, simulando o processo de atribuição prática de story points. Para avaliação, foi calculado o MAE.

4 Resultados

Os resultados indicaram que, embora todos os modelos apresentassem alguma capacidade preditiva, o fine-tuned com cross-project (ou seja, o treino com todos os projetos do conjunto de dados) superou os demais métodos para a maioria dos projetos (Tabela 2). O MAE obtido pelo modelo fine-tuned foi significativamente mais baixo na maioria dos projetos, sugerindo um erro menor nas estimativas de conclusão dos projetos. Algumas comparações entre as métricas foram realizadas e são descritas a seguir:

Few Shot x Zero Shot: Quando comparado o modelo few shot com o modelo zero shot, o modelo few shot performou melhor na maioria dos projetos, 10 de 16. Enquanto o modelo zero shot teve MAE melhor (ou seja, menor) em somente 6 de 16 projetos (Figura 3).

Few Shot x Fine-tuning: Quando comparado o modelo Few shot com o modelo Fine-tuning, o modelo Fine-tuning performou melhor na maioria dos projetos, 14 de 16. Enquanto o modelo Few Shot performou melhor em 2 de 16 projetos. Veja a Figura 4.

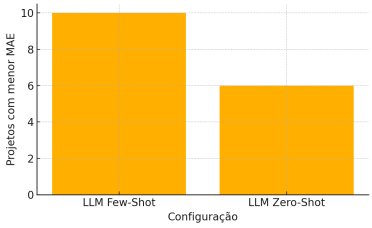


Figura 3: Desempenho: Few-shot vs Zero Shot

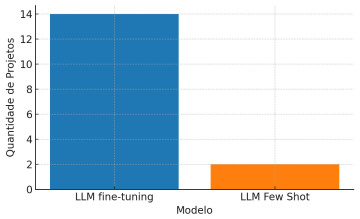


Figura 4: Desempenho: Fine-tuning vs Few-Shot

Few Shot x Fine-tuning x TF-IDF RL: Conforme a Figura 5, quando comparado o modelo fine tuning, com few shot e com TF-IDF-RL, o modelo Fine-tuning performou melhor na maioria dos projetos, 11 de 16. TF-IDF com RL 4 de 16 e few shot 1 de 16.

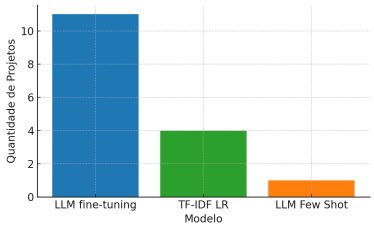


Figura 5: Desempenho: Todos

A baixa performance do modelo few-shot em alguns domínios (ex.: P9) destaca sua limitação em generalização fora do domínio de treino. A performance consistente do modelo fine-tuning em 14 dos 16 projetos sugere que ele possui maior robustez na generalização entre diferentes domínios, mesmo sendo treinado em regime cross-project. No projeto P14, observou-se um aumento abrupto no MAE do modelo TF-IDF + RL, atingindo um valor atípico de 308.49. Uma análise qualitativa das user stories revelou uma predominância de descrições extremamente curtas, o que pode ter dificultado a extração de boas features pela abordagem baseada em frequência de palavras.

Tempo e Custo: o modelo fine-tuning exigiu tempo de ajuste inicial (aproximadamente 18 minutos com GPU A100 no Google Colab), mas apresentou tempo de inferência médio de apenas alguns segundos por user story. Considerando o valor médio (no período em que foi realizado o estudo no ano de 2025) de US$ 0,56 por hora de uso de GPU A100 em ambientes como Google Colab Pro ou AWS, o custo de treinamento foi inferior a US$ 0,20. Embora

Model	P1	P2	P3	P4	P5	P6	P7	P8	P9	P10	p11	p12	p13	p14	p15	p16
LLM few shot	5.98	7.07	2.12	5.65	7.36	3.05	4.63	2.64	14.50	6.48	5.50	4.51	5.62	**2.56**	4.05	2.33
LLM fine-Tuning	**1.81**	**3.09**	2.19	**3.79**	**5.42**	2.04	2.38	**1.52**	**8.44**	**2.38**	**3.70**	**1.96**	**3.36**	3.66	**2.64**	2.10
LLM Zero Shot	3.19	3.57	7.00	7.57	7.86	7.59	6.13	6.01	10.06	4.58	4.24	5.23	6.31	7.67	3.81	6.19
TF-IDF LR	6.49	3.89	**1.07**	4.93	30.71	**1.31**	**2.05**	2.30	17.10	2.94	5.06	9.26	5.63	308.49	6.45	**1.34**

Tabela 2: MAE dos Modelos Utilizados (menores valores em negrito)

exija um investimento inicial em hardware compatível, apresenta custo marginal muito mais baixo para inferência em larga escala, podendo ser executado localmente em ambientes com recursos limitados. Por outro lado, os modelos few-shot e zero-shot, apesar de dispensarem treinamento prévio, dependem de múltiplas chamadas a LLMs via prompt, o que resulta em um custo acumulado mais elevado por predição/inferência, especialmente quando se utilizam APIs comerciais. Assim, em contextos onde o número de inferências é elevado e a infraestrutura para treinamento está disponível, o modelo fine-tuning se mostra mais vantajoso também sob a perspectiva econômica, além de entregar melhor acurácia.

5 Ameaças à Validade

Validade Interna: a) existe uma assimetria no desenho experimental. O modelo fine-tuning foi treinado com dados cross-project, enquanto os outros modelos (few-shot, zero-shot, TF-IDF) foram treinados por projeto. Isso pode introduzir uma vantagem artificial ao fine-tuning devido à maior diversidade e volume de dados durante o treinamento. b) ausência de validação cruzada. O estudo utilizou uma divisão fixa (70% 30%) dos dados. Isso pode levar a resultados enviesados caso a divisão não represente adequadamente a distribuição dos dados. c) O prompt utilizado foi escrito em português, enquanto o modelo ajustado via fine-tuning foi treinado com instruções (antes do processo de tokenização) em inglês. Essa diferença linguística pode ter influenciado significativamente o desempenho dos modelos, já que a formulação do prompt tem impacto direto na qualidade da resposta gerada pelos LLMs, podendo introduzir viés nos resultados [18]. **Validade Externa**: Dependência de um único dataset (Deep-SE): Os experimentos foram realizados apenas com o conjunto de dados Deep-SE. A generalização para outros contextos de desenvolvimento ágil, com características textuais diferentes, não foi avaliada. Isso limita a aplicabilidade dos resultados a outros domínios. **Validade de Construto**: a) conversão de valores contínuos para sequência de Fibonacci. O arredondamento do valor contínuo predito para um número da sequência de Fibonacci pode introduzir distorções artificiais na métrica de erro, pois aproximações distintas podem resultar no mesmo valor final. b) dependência do MAE como única métrica de avaliação. Embora o MAE seja uma métrica informativa, o uso exclusivo dela pode não capturar aspectos como dispersão dos erros ou sensibilidade a outliers. **Validade de estatística**: Por fim, apesar de apresentar comparações de MAE entre modelos, não há aplicação de testes estatísticos para verificar se as diferenças observadas são estatisticamente significativas.

6 Trabalhos Relacionados

Estudos recentes têm reforçado o uso de modelos de linguagem de larga escala para estimativa de esforço em projetos ágeis. Um deles é GPT2SP [6], uma abordagem baseada no modelo GPT-2 ajustado, demonstrando ganhos significativos sobre modelos clássicos como regressão linear e random forest, embora restrita ao fine-tuning supervisionado e sem exploração das estratégias zero-shot ou few-shot. Outro estudo utilizou redes neurais com mecanismos de atenção, superando modelos SVM, mas apresentando limitações de generalização entre projetos [7]. Em consonância com esses avanços, o presente estudo amplia a análise ao comparar diferentes paradigmas de aplicação de LLMs, incluindo fine-tuning, few-shot e zero-shot, em um mesmo conjunto de dados consolidado. Além do desempenho preditivo, também são consideradas variáveis práticas como custo computacional e reprodutibilidade, oferecendo uma avaliação mais abrangente dos LLMs na estimativa.

7 Considerações Finais

Foi investigada a eficácia de LLMs na estimativa de story points a partir do texto e descrição de user stories. Os resultados desta investigação sugerem que o Fine-tuning possui potencial significativo para melhorar a precisão na previsão de story points em projetos ágeis. Essa maior precisão pode auxiliar no planejamento e gerenciamento de projetos, mitigando problemas como vieses humanos e variação entre equipes frequentemente associados à estimativa tradicional baseada em consenso. A abordagem de utilizar um modelo ajustado (fine-tuning) com todos os projetos de um conjunto de dados (cross-project) pareceu contribuir para sua superioridade em relação aos modelos treinados individualmente por projeto para as técnicas baseline. Como trabalhos futuros, é sugerida a validação do modelo ajustado com dados de outros conjuntos de dados, como, por exemplo, o NeoDataset [12] usado em Neo et al. [11]. Também é recomendada a avaliação de outros modelos LLM GPT com pesos disponíveis (open weight) para ajuste, como Qwen3, da empresa Alibaba, LLaMA da Meta, entre outros, além da utilização de técnicas mais robustas como cross-validation e amostra estratificada para as métricas.

DISPONIBILIDADE DE ARTEFATO

O código-fonte está disponível em *https://github.com/giseldo/artigo-storypoint-deep-se-llm*. O dataset em *https://huggingface.co/datasets/giseldo/deep-se*. O modelo ajustado em endereço *https://huggingface.co/giseldo/distilbert_bert_uncased_finetuned_story_point*.

AGRADECIMENTOS

O presente trabalho foi realizado com apoio da Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Código de Financiamento 001.

REFERÊNCIAS

[1] N Akhila et al. 2023. Comparative study of bert models and roberta in transformer based question answering. In *2023 3rd International Conference on Intelligent Technologies (CONIT)*. IEEE, 1–5.

[2] Levi Alexander and Riyanto Jayadi. 2024. Machine Learning for Story Point Estimation: Do Large Language Models Outperform Traditional Methods? *Journal of Theoretical and Applied Information Technology* 102, 20 (2024), 7387–7399.

[3] Morakot Choetkiertikul, Hoa Khanh Dam, Truyen Tran, Aditya Ghose, and John Grundy. 2015. Predicting Delivery Capability in Iterative Software Development. *JOURNAL OF LATEX CLASS FILES* 14, 8 (2015), 551–573. doi:10.1109/TSE.2017.2693989

[4] Morakot Choetkiertikul, Hoa Khanh Dam, Truyen Tran, Trang Pham, Aditya Ghose, and Tim Menzies. 2019. A Deep Learning Model for Estimating Story Points. *IEEE Transactions on Software Engineering* 45, 7 (2019), 637–656. doi:10.1109/TSE.2018.2792473 arXiv:1609.00489

[5] Mike Cohn. 2005. *Agile Estimating and Planning*.

[6] M Fu and C Tantithamthavorn. 2023. GPT2SP: A Transformer-Based Agile Story Point Estimation Approach. *IEEE Transactions on Software Engineering* 49, 02 (2023), 611–625. doi:10.1109/TSE.2022.3158252

[7] Haithem Kassem, Khaled Mahar, and Amani A. Saad. 2023. Story Point Estimation Using Issue Reports With Deep Attention Neural Network. *E-Informatica Software Engineering Journal* 17, 1 (2023), 1–15. doi:10.37190/e-Inf230104

[8] William B. Langdon, Javier Dolado, Federica Sarro, and Mark Harman. 2016. Exact Mean Absolute Error of Baseline Predictor, MARP0. *Information and Software Technology* 73 (2016), 16–18. doi:10.1016/j.infsof.2016.01.003

[9] Shervin Minaee, Tomas Mikolov, Narjes Nikzad, Meysam Chenaghlu, Richard Socher, Xavier Amatriain, and Jianfeng Gao. 2025. Large Language Models: A Survey. *arXiv* (2025). arXiv:2503.23037 http://arxiv.org/abs/2503.23037

[10] Giseldo da Silva Neo, Antão B. Moura, Alana Viana Borges da Silva, Neo, and Evandro de Barros Costa. 2025. A Predictive Model for Story Points leveraging features like readability and sentiment from User Story description. *ITS2025 - Intelligent Tutoring Systems* (2025).

[11] Giseldo da Silva Neo, José Antão Beltrão Moura, Hyggo Almeida, Alana Viana Borges da Silva Neo, and Olival de Gusmão Freitas. 2024. User Story Tutor (UST) to Support Agile Software Developers. *International Conference on Computer Supported Education, CSEDU - Proceedings* 2 (2024), 51–62. doi:10.5220/0012619200003693 arXiv:2406.16259

[12] Giseldo da Silva Neo, Alana Viana Borges da Silva Neo, Kleber Jose Araújo Galvão Filho, José Antão Beltrão Moura, and Olival de Gusmão Freitas Junior. 2024. NeoDataset: um conjunto de dados com user stories e story points. *Revista dos Mestrados Profissionais* 133, 2 (2024), 194–211.

[13] Bodem Niharika and Shivali Chopra. 2024. Story Point Estimation Using Machine Learning for Agile Projects. *SSRN Electronic Journal* (2024). doi:10.2139/ssrn.4485276

[14] Simone Porru, Alessandro Murgia, Serge Demeyer, Michele Marchesi, and Roberto Tonelli. 2016. Estimating story points from issue reports. *ACM International Conference Proceeding Series* (2016). doi:10.1145/2972958.2972959

[15] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2018. Language Models are Unsupervised Multitask Learners. (2018).

[16] Sebastian Raschka. 2024. *Build a Large Language Model (From Scratch)*. Simon and Schuster.

[17] Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. 2019. DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter. (2019), 2–6. arXiv:1910.01108 http://arxiv.org/abs/1910.01108

[18] E. G. Santana, Gabriel Benjamin, Melissa Araujo, Harrison Santos, David Freitas, Eduardo Almeida, Paulo Anselmo da M. S. Neto, Jiawei Li, Jina Chun, and Iftekhar Ahmed. 2025. Which Prompting Technique Should I Use? An Empirical Investigation of Prompting Techniques for Software Engineering Tasks. (2025). arXiv:2506.05614 http://arxiv.org/abs/2506.05614

[19] Federica Sarro, Alessio Petrozziello, and Mark Harman. 2016. Multi-objective software effort estimation. *Proceedings - International Conference on Software Engineering* 14-22-May- (2016), 619–630. doi:10.1145/2884781.2884830

[20] Ezequiel Scott and Dietmar Pfahl. 2018. Using developers' features to estimate story points. *ACM International Conference Proceeding Series* 106 (2018), 106–110. doi:10.1145/3202710.3203160

[21] Martin Shepperd and Steve MacDonell. 2012. Evaluating prediction systems in software project estimation. *Information and Software Technology* 54, 8 (2012), 820–827. doi:10.1016/j.infsof.2011.12.008

[22] Krishnamoorthy Srinivasan and Douglas Fisher. 2005. Machine Learning Approaches to Estimating Software Development Effort. *Machine Learning Applications In Software Engineering* 21, 2 (2005), 52–63.

[23] Jeff Sutherland. 2014. *A arte de fazer o dobro do trabalho na metade do tempo* (1 ed.).

[24] Ritesh Tamrakar and Magne Jørgensen. 2012. Does the use of Fibonacci numbers in planning poker affect effort estimates? *IET Seminar Digest* 2012, 1 (2012), 228–232. doi:10.1049/ic.2012.0030

[25] Vali Tawosi, Rebecca Moussa, and Federica Sarro. 2022. Agile Effort Estimation: Have We Solved the Problem Yet? Insights From A Replication Study. *IEEE Transactions on Software Engineering* (2022), 1–19. doi:10.1109/TSE.2022.3228739 arXiv:2201.05401

[26] Victor Uc-Cetina. 2023. Recent Advances in Software Effort Estimation using Machine Learning. (2023), 1–10. arXiv:2303.03482 http://arxiv.org/abs/2303.03482

[27] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention Is All You Need. *CoRR* abs/1706.03762 (2017). http://arxiv.org/abs/1706.03762

[28] Raul Sidnei Wazlawick. 2009. *Metodologia de pesquisa para ciência da computação*. Vol. 2. Elsevier Rio de Janeiro.

[29] Burcu Yalçıner, Kıvanç Dinçer, Adil Gürsel Karaçor, and Mehmet Önder Efe. 2024. Enhancing Agile Story Point Estimation: Integrating Deep Learning, Machine Learning, and Natural Language Processing with SBERT and Gradient Boosted Trees. *Applied Sciences (Switzerland)* 14, 16 (2024). doi:10.3390/app14167305

Apêndice H

Writing Better *User Stories* and Estimates *Story Point* with Machine Learning and Natural Language Processing

Artigo publicado no Journal *Springer Nature Computer Science* [29].

SN Computer Science (2025) 6:821
https://doi.org/10.1007/s42979-025-04363-w

ORIGINAL RESEARCH

Writing Better User Stories and Estimates Story Point with Machine Learning and Natural Language Processing

Giseldo da Silva Neo[1,2] · José Antão Beltrão Moura[2] · Hyggo Oliveira de Almeida[2] · Alana Viana Borges da Silva Neo[2,4] · Olival de Gusmão Freitas Júnior[3]

Received: 18 January 2025 / Accepted: 1 September 2025

Abstract

User Stories record what must be built in projects that use agile practices and serve both to estimate effort, generally measured in Story Points, and to plan what should be done in a Sprint. Therefore, it is essential to create simple, easily readable, and comprehensive User Stories. This article proposes a method to help agile teams create high-quality User Stories. The proposal uses natural language processing, large-scale language models, and machine learning to provide personalized recommendations to improve User Stories, estimate effort in Story Points, and evaluate text readability. The tool was evaluated with a questionnaire based on the Technology Acceptance Model and AttrakDiff frameworks and measured usability, ease of use, external factors, and participants' attitudes toward the proof-of-concept. The proposal demonstrates the potential to improve user story creation, and the participants responded positively, describing it as both useful and easy to use. Furthermore, the data set used is also presented.

Keywords User story · Story points · Effort estimation · Recommendation systems · Readability index

Introduction

Each year, Standish Group International publishes a report that consolidates data from technology projects conducted by companies in various countries known as the Chaos Report [1]. The data in this report suggest that fewer than one third of the software technology projects surveyed were fully successful. In this report, most of the projects failed to meet estimated timelines and budget constraints.

Maybe this happens because software development involves numerous variables and is susceptible to various failures [2]. A significant portion of these failures comes from issues related to requirements specification and other contributing factors. To mitigate these challenges, methodologies and frameworks such as Agile methods offer a conceptual structure to guide software engineering projects.

A User Story is a brief and straightforward statement that describes a feature from the perspective of the user who desires it, and is used to define the scope of a software project [3]. This technique for requirement analysis captures the "who", "what", and "why" concisely and simply, typically providing just enough detail to be written by hand on a small note card. The simplicity and brevity of User Stories enable quick understanding and flexibility in development. The unit

José Antão Beltrão Moura, Hyggo Oliveira de Almeida, Alana Viana Borges da Silva Neo, and Olival de Gusmão Freitas Júnior have contributed equally to this work.

✉ Giseldo da Silva Neo
giseldo.neo@ifal.edu.br

José Antão Beltrão Moura
antao@computacao.ufcg.edu.br

Hyggo Oliveira de Almeida
almeida@computacao.ufcg.edu.br

Alana Viana Borges da Silva Neo
alana.neo@ifms.edu.br

Olival de Gusmão Freitas Júnior
olival@ic.ufal.br

1 Computing Coordination, Federal Institute of Alagoas, Viçosa, Alagoas, Brazil

2 Center for Electrical and Computer Engineering, Federal University of Campina Grande, Campina Grande, Paraíba, Brazil

3 Institute of Computing, Federal University of Alagoas, Maceió, Alagoas, Brazil

4 Directorate of Education and Postgraduate Studies, Federal Institute of Mato Grosso do Sul, Corumbá, Mato Grosso do Sul, Brazil

Published online: 12 September 2025

of measurement usually used along with User Stories is the Story Point.

A Story Point is a unit of measurement commonly used in agile methodologies to estimate the relative effort required to complete a task or feature, typically described as a User Story. For example, if User Story A is assigned 2 points and User Story B is assigned 5, it indicates that B is perceived as more challenging or demanding than A, although not necessarily 2.5 times longer in duration. Teams often use nonlinear scales such as the Fibonacci sequence (1, 2, 3, 5, 8, 13...) to avoid overly precise estimates and to encourage relative comparisons between tasks. Story points do not directly measure time. Instead, they reflect a combination of three key dimensions: technical complexity, amount of work, and uncertainty or risk involved in completing the task [3].

These User Stories are generally stored in software tools that manage the entire project life cycle, aiding in the organization and tracking of requirements [4] (e.g. Jira and GitLab). By analyzing the data recorded in these tools, valuable insights can be derived for various research efforts in software engineering, providing empirical data to improve methodologies and practices [5].

However, crafting a well-defined User Story can be challenging. A User Story may lack sufficient detail, making it difficult to grasp the expected outcome, or conversely, it may be overly comprehensive. For instance, a stakeholder might misjudge the level of detail required and outline the scope of an entire module or system, which is not appropriate for a User Story. Moreover, the quality of User Stories significantly affects the agile team's ability to make accurate estimates. The writing of effective user stories, which serve as the basis for the estimation of effort, has been identified as one of the top five most critical issues facing agile teams, according to feedback from 119 developers [6].

In this context, how can agile teams support the construction of User Stories and assist in the development process during both the User Story preparation phase and task effort estimation? And how to apply natural language processing techniques to enhance the User Story creation process and provide more accurate Story Point estimates? In a more broader manner, the research problem is how to improve the quality of User Stories and Story Point estimation in agile projects.

Thus, the goal of this study is to enhance the User Story creation process by recommending improvements through the application of natural language processing and automating the estimation of Story Point. The hypothesis is that the proposed approach can assist agile teams in constructing higher-quality User Stories for better story point estimation. The expected contribution is to provide development teams that follow agile practices with tools to create more effective User Stories.

As a proof of concept, we designed a web application, called User Story Tutor (UST). It takes a User Story text as input and provides personalized recommendations for improving it, powered by a large language model (LLM). LLMs are advanced deep learning models pre-trained on vast datasets, enabling them to generate contextually aware suggestions. In addition to improvement recommendations, the tool offers an effort estimate in Story Points, generated by a machine learning algorithm trained on data from previous projects. The application also displays readability indexes for the User Story, serving as an indicator of the clarity and ease of understanding of the text.

We used the Design Science Research methodology, which consists of three phases: problem identification, solution design, and evaluation, to develop the proposed tool [7]. For the evaluation phase, a survey was conducted using the Technology Acceptance Model framework (TAM) [8] and the AttrakDiff evaluation framework [9]. A total of 40 agile practitioners, who were not involved in the development of the Web app, evaluated the proposed solution and responded to the survey. The results of the TAM and AttrakDiff evaluations indicate that the tool successfully meets its objectives, receiving positive acceptance from the participants.

The remainder of this paper is organized as follows. Section "Background" provides an overview of the key concepts necessary for understanding the following sections. Section "Materials and Methods" describes the methodology and the methodological artifacts adopted in the study and the technical details of the proposed solution. Section "Results and Discussion" discusses the evaluation results. Section "Related Work" reviews and compares related work. Section "Limitations and Threats" examines threats to the validity of our investigation and our findings. Finally, Section "Conclusion" concludes with final considerations.

Background

User Stories and Story Points

The requirements in agile are typically captured in User Stories, which describe the tasks the development team will estimate and implement [10]. They are written in natural language to facilitate communication and understanding among team members and are usually measured as Story Points.

User stories play a central role in the development of requirements for teams that employ agile methodologies. A key characteristic of agile methods is their emphasis on rapid, value-driven deliveries within short time frames while adapting to changes as quickly as possible [11]. This approach has proven effective and has been widely adopted

in project management [12], both in industry [13, 14] and government sectors [15].

The quality of a user story is commonly defined by how well it adheres to essential attributes such as clarity, estimability, controllability, and testability. These characteristics reflect whether a user story is understandable to stakeholders, can be estimated for effort, is manageable, and can be validated through testing [16].

Most teams that use User Stories, or agile methodologies in general, also rely on software tools to manage their projects and, most importantly, to keep a record of their User Stories [4]. By analyzing the data captured by these tools, valuable insights can be extracted for various software engineering research efforts, including studies aimed at improving the quality of User Stories themselves [17].

GitLab is one of the management tools commonly used by agile teams to track and record User Stories [18]. It enables software engineers to automate numerous tasks throughout the development cycle, including the creation and modification of User Stories [19]. In GitLab, a User Story is documented as an Issue, and various details are stored for each User Story, such as the title, task description, and its effort estimate in Story Points.

The data stored in these management tools (e.g. Jira or GitLab) can support decision making in various software engineering scenarios, including the assignment of User Stories [20], improving the quality of User Story descriptions [21], iteration planning [22], sentiment analysis of developers writing User Stories [23–25], effort estimation for User Stories [26–30], and prioritization of User Stories [31–33].

Recommendation Systems with LLM

Recommendation systems are software tools and techniques designed to suggest items that are likely to be of interest to a specific user [34]. In the context of learning, recommendations play an important role by helping teachers and students discover content that is better aligned with their profiles and needs, thereby enhancing the learning process.

Recommendation systems have gained significant popularity in the educational field, providing various types of recommendations to students, teachers, and schools. These systems help reduce information overload for students by providing the right information at the right time and in the appropriate format, tailored to the student's interests [35].

The evolution of Recommendation Systems is shifting towards the use of hybrid techniques, which combine two or more distinct recommendation methods to overcome individual limitations and leverage the strengths of each [36]. A hybrid Recommendation System refers to any system that integrates multiple recommendation techniques to generate more accurate output. The primary types of recommendation techniques include collaborative, content-based, demographic, knowledge-based methods, and hybrids [37].

More recently, Large Language Models (LLMs), such as ChatGPT from OpenAI or Gemma from Google, have gained attention for their ability to generate sophisticated recommendations. These models are pre-trained on vast amounts of diverse data, enabling them to understand context, generate human-like text, and provide tailored suggestions across various domains. This makes LLMs particularly useful in recommendation systems, as they can analyze complex inputs, adapt to diverse user needs, and offer highly personalized recommendations.

Other study proposed an automated approach to estimating software size at the early stage of development using the COSMIC method [38]. The authors introduce RE-BERT, a BERT model pre-trained specifically for requirements engineering texts, and combine it with deep learning algorithms such as a Multi-Layer Perceptron and BERT-based regressors. The experiments show that RE-BERT MLP delivers better accuracy in size estimation than generic BERT models. The study demonstrates the effectiveness of domain-specific pre-trained models for regression tasks in software engineering, overcoming the challenges of manual COSMIC measurement. The results suggest that this approach can support automated software size measurement even with informal requirements, and the work points to future improvements and applications to other software engineering tasks.

Although an LLM model excels at generating contextualized responses, it is particularly suited for applications where absolute reliability is not mission-critical, such as assisting with creative tasks, content generation, or general-purpose recommendations [39]. However, the flexibility of LLMs opens the door to a wide range of use cases, including education, e-commerce, entertainment, and more.

The ability of these models to process natural language at scale allows for more dynamic and nuanced recommendation systems that go beyond traditional rule-based methods. Despite some limitations in using LLMs for recommendations, such as unreliable responses, a limited context window, and the inability to learn from interactions, they can still be highly effective in specific domains [39]. They are particularly suited for scenarios with well-defined intentions because of their structured nature and predictable output.

Readability Indexes for Natural Language

A User Story is typically written in natural language, which makes its clarity and readability crucial for effective communication among stakeholders. To assess the

readability of a User Story, we can apply readability indexes. These indexes are widely used in the literature to predict the ease with which a text can be read and understood. Although readability indices are commonly applied in fields such as education and linguistics, they have also found applications in computer science, such as the detection of fake news [40], where text clarity plays a role in distinguishing deceptive content.

They have been used by educators since the 1920s, and by 1980, more than 200 different formulas for calculating readability were already known [41]. Despite receiving criticism from researchers who highlighted their limitations [42], empirical studies have confirmed a correlation between these indices and the actual readability of a text [43].

Readability indices provide a numerical indicator of how easy or difficult a text is to read for others [41]. They are calculated using algorithms that often take into account various factors such as the number of words, characters, sentences, syllables, and the presence of complex words in the text.

The Gunning Fog Index (GFI) is one of the most widely used and studied readability indices, frequently applied to analyze text [43]. Developed by Robert Gunning in 1954, the Fog Index assigns a numerical value to a text based on factors such as word count, sentence length, and the percentage of complex words. The higher the index value, the more difficult the text is to read. Complex words are typically defined as those that contain three or more syllables. As the proportion of complex words increases, so does the difficulty of reading the text, resulting in a higher index score and lower readability.

The formula for calculating the Fog Index is well documented [44] and is calculated by adding the average sentence length to the percentage of complex words, then multiplying the sum by 0.4, as shown in Table 1.

Another widely used readability index is the Flesch Reading Ease (FRE) score [46]. This index measures how difficult a text is to read, with higher values indicating easier readability. The maximum score is 121.22, while there is no minimum value, which also allows for negative scores. The calculation of the Flesch Reading Ease is shown in Table 1.

Table 1 Readability index formulas. Adapted from [45]

Index	Formula
GFI	$0.4 \cdot \left[\left(\frac{words}{sentences}\right) + 100 \cdot \left(\frac{words\ complex}{words}\right)\right]$
FRE	$206.835 - 1.015 \cdot \left(\frac{words}{sentences}\right) - 84.6 \cdot \left(\frac{syllables}{words}\right)$
CLI	$0.0588 \cdot L - 0.296 \cdot S - 15.8$
ARI	$4.71 \cdot \left(\frac{characters}{words}\right) + 0.5 \cdot \left(\frac{words}{sentences}\right) - 21.43$

This test, one of the oldest and most commonly applied, is based on two factors: the longer the average sentence length, the more difficult the text is to read; and the greater the average number of syllables per word, the harder the text becomes. A higher score reflects better readability and ease of comprehension.

The Coleman–Lieu index is another readability metric [46], calculated using a different formula, as shown in Table 1. In this formula, L represents the average number of letters per 100 words and S denotes the average number of sentences per 100 words.

Finally, the Automated Readability Index (ARI) is another commonly used readability metric. It is designed to estimate the level of reading that is required to understand a text. The ARI is computed using a formula which takes into account both the number of characters per word and the average sentence length. This calculation is shown in Table 1. Unlike other readability indexes that rely on syllable counts, the ARI focuses on the total number of characters, which makes it particularly useful for analyzing digital text where counting characters is more straightforward than counting syllables.

In the agile context, they can provide quantitative measures of readability, helping teams gauge whether a User Story is easily understandable or may need revision to improve clarity and comprehension. Using such tools, agile teams can ensure that their User Stories are accessible to all stakeholders, improving communication, and reducing misunderstandings during the development process.

Materials and Methods

The research methodology used in this research was Design Science Research [7]. We designed a web application that uses Natural Language Processing, Readability Indexes, and Machine Learning Prediction as a proof-of-concept to improve user story writing. We used a survey, supported by a Google Forms questionnaire, to evaluate the proposed solution. The development was carried out in the stages presented in Figure 1.

Validation To evaluate our proposal, we conducted a survey using the Technology Acceptance Model (TAM) framework [8] and the AttrakDiff evaluation framework [9]. TAM is a theory of information systems that models how users accept and adopt technology. For statistical analysis within the TAM framework, the Cronbach alpha was used to assess reliability. The AttrakDiff test [9] evaluates both hedonic and pragmatic quality factors, providing a more comprehensive assessment of the proposal and complementing the TAM framework. The survey gathered participants' perceptions and suggestions regarding the

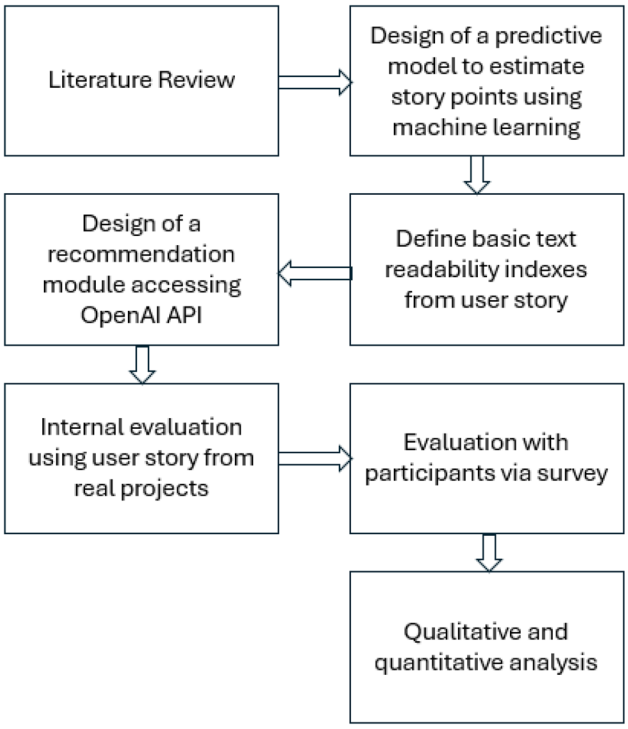


Fig. 1 Step by step of the procedures performed

proposal, which is designed to assist agile practitioners in improving the quality of User Stories.

Proposal: The proposed solution is divided into three modules: Recommender, Estimator, and Readability. The architecture of the application is illustrated in Fig. 2. The **recommender** (C) leverages an LLM to recommend improvements, the **readability** (B) displays indexes extracted by mathematical formulas and the **estimates** (A) the effort to get story points with a machine learning predictor. For each module, specific procedures were selected and executed. The prototype screen (Fig. 3) allows a developer of the agile team to input the User Story text. Any User Story from a real project can be used. After entering the User Story description in text format, the developer clicks on the "Analyze" button. Then it initiates the necessary processes that activate the responses of the existing modules.

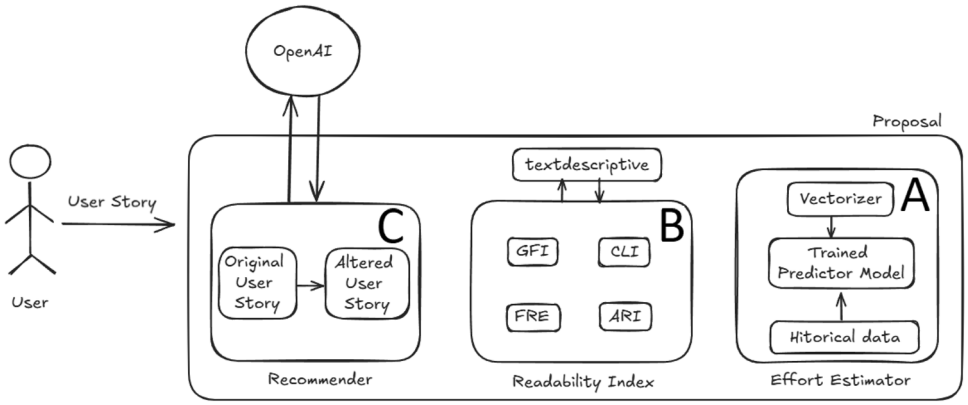
Technologies: For development, Python was chosen as the programming language, given its growing popularity, especially in machine learning applications [47]. For the interface, StreamLit was used, an open source library to build applications in machine learning and data science [48]. The Recommender module communicates with the OpenAI LLM GPT-3.5 turbo via an API query, while the scikit-learn library was used for machine learning tasks in the estimator module [49]. The entire source code for the project is available on GitHub and the web application can be tested on StreamLit Cloud.

Interface: The prototype screen (Fig. 3) allows a developer of the agile team to input the User Story text. Any User Story from a real project can be used. After entering the User Story description in text format, the developer clicks on the "Analyze" button. Then it initiates the necessary processes that activate the responses of the existing modules.

Recommender Module

This module (C in Fig. 3) is responsible for suggesting improvements to User Stories by generating text in natural language. The LLM model used is pre-trained in vast amounts of text sourced from the Internet [50]. However, customization of the prompt is necessary to personalize the output more effectively. To tailor the response from the OpenAI LLM model, we send a carefully crafted prompt that aligns with the specific requirements of the recommendation system. Following OpenAI's guidelines for creating effective prompts, three key techniques were employed in the prompt design. This module handles User Story recommendation requests by managing the prompt, combining it with the new User Story text, querying the LLM, and preparing the response for user presentation (Refer to Table 2 for the prompt design techniques employed in this study)).

Fig. 2 High-level architecture of the User Story Tutor application



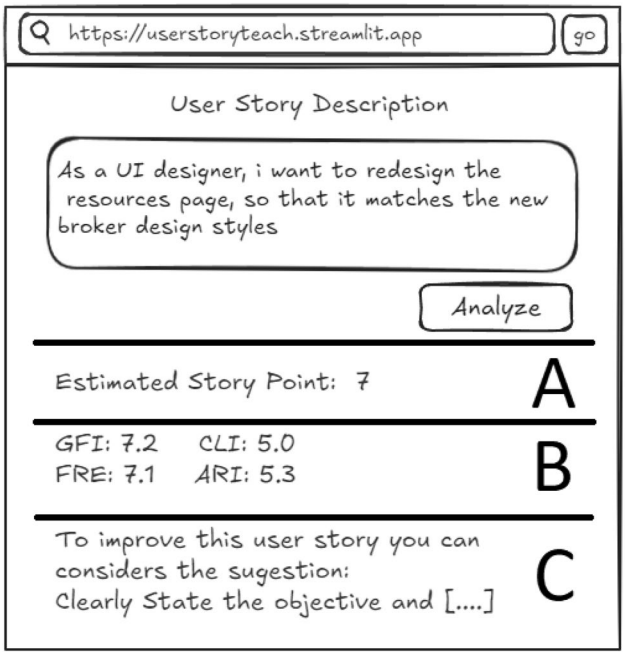


Fig. 3 Proposal Interface prototype

The recommender module provides suggestions based on the User Story description entered by the developer. For this, the model used is OpenAI's GPT-3.5-turbo. The parameters utilized by the recommender module are sent via a hidden prompt, as detailed in Fig. 4. These parameters were subjected to a refinement process similar to the creation of a search query in a systematic review of the literature, being revised and adjusted until the final version was reached, with the support of the dataset described in Section "Data Set".

Readability Module

For the User Story readability module (B in Fig. 3), the readability indexes of the text are extracted using the textdescriptive [51] library. Next, it shows the index on the screen with the "metrics" component from StreamLit. The purpose of the readability module is to allow the creator of the User Story to see some quantitative measure of how easy the text of their User Story is to read, and compare after some changes using the recommender module.

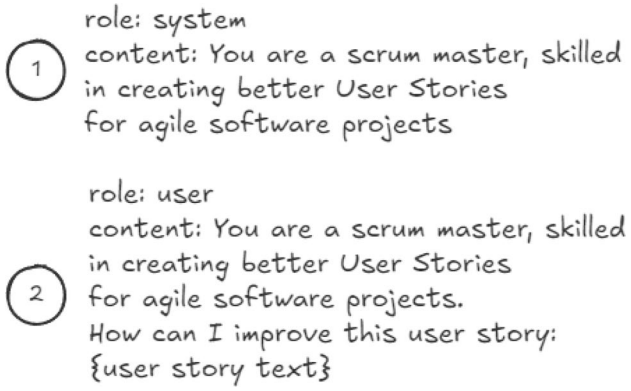


Fig. 4 OpenAI personalized prompt

Calculate four text readability indexes: Gunning Fog, Automated Readability Index, Coleman-Liau Index, and Flesch Reading Ease (see Section 2 for more details). To simplify the interpretation of these readability scores, a variable called the Final Result was created, representing the arithmetic mean of the four selected indexes and presented in the web application.

Estimate Module

This module (A in Fig. 3) predicts Story Points using a pre-trained model based on historical data, providing a reference to assist developers with their effort estimations. Lastly, the Readability module extracts readability indexes from the text using basic natural language processing techniques.

The estimator module employs a supervised learning algorithm, chosen for its ability to provide the most accurate predictions of Story Points based on the input User Story text. The development of this module followed a structured approach, including data collection, data exploration, data preparation, model creation, training, validation, hyperparameter tuning, and final implementation.

The Effort Estimation module provides an effort estimate in Story Points based on the User Story description. The predictive model and vectorizer are loaded using the Joblib library, with the Support Vector Regression (SVR) algorithm selected for prediction. The User Story text is transformed into a bag-of-words representation using the

Table 2 Prompt design guidelines. Adapted from [45]

Technique	Description
Clarity in instruction	We seek a clear and precise prompt to not generate doubts when returning the recommendation. The probability of a good return recommendation depends on the objectivity of the hidden prompt sent along with the recommendation
Split complex tasks into simpler tasks	Intending to limit the task, we sent (a prompt) text to limit the set of return possibilities, as complex tasks generally have a higher error rate than simpler task requests
Validate prompt	Several interactions were necessary to create the prompt used in search of improvements and following the good practices recommended by OpenAI itself

term frequency - inverse document frequency (TF-IDF) technique. In production, both the vectorizer and model are loaded into memory for efficient prediction. Once the User Story text is transformed into a matrix by the vectorizer, it is passed to the model, which generates the Story Points estimate. The loaded model was trained using data from previous open source GitLab projects.

The User Story Estimator was trained using User Stories from various agile projects. This data set was sourced from real open source projects available in an open source repository. More details about the data set are provided in Section 3.4. The metric used to select the best algorithm was the Mean Absolute Error (MAE) with cross-validation. Once the optimal model was identified, it was trained in the entire dataset and integrated into the proposed application. The MAE is calculated according to Eq. 1. $E_{current}$ is the current effort, $E_{predicted}$ is the predicted effort, and n is the number of observations [46].

$$MAE = \frac{1}{n}\sum_{i=1}^{n}\left|E_{actual} - E_{predicted}\right| \quad (1)$$

The Mean Absolute Error (MAE) was chosen as the model evaluation metric due to its interpretability and robustness in regression tasks with noisy data, such as effort estimation in Story Points. Unlike RMSE, which penalizes larger errors more heavily due to the squared differences, MAE provides a more balanced assessment that is less sensitive to outliers, directly reflecting the average absolute error in the predicted variable's units. This characteristic is particularly advantageous in the context of this study, which aims to provide a practical and understandable estimate for agile teams, avoiding distortions caused by atypical cases. Moreover, MAE aligns with the regression-based approach adopted in this work, using continuous values, and maintains methodological consistency with the prediction model trained using SVR and TF-IDF. The task of estimating Story Points from a User Story description was treated as a regression problem, since the goal is to predict a continuous numerical value that represents the estimated effort, rather than classifying the input into discrete categories. For this reason, classification-based metrics, such as precision, recall, and F1 score, are not suitable for evaluating model performance in this context. These metrics are designed for problems where the outputs are class labels, whereas in this study the focus is on minimizing the difference between actual and predicted values, which is better captured by continuous error metrics such as MAE. Thus, the use of regression techniques and their corresponding metrics ensures greater methodological consistency and alignment with the proposed problem.

Data Set

For the estimator module, we consolidate a dataset to train the machine learning predictor. This data set (aka NEODATASET) includes data from 33 software development projects, containing 12.2627 User Stories sourced from GitLab open-source repositories with a total of 2.0474 Story Points. The data set is stored in the coma-separated values (CSV) format, given the simplicity of dealing with this format, and is publicly accessible on Mendeley Data and in Hugging Face, allowing the broader community to contribute, similar to how other open data sets are managed. A glimpse of the data set can be seen in Fig. 5.

This data set was collected between January 2023 and April 2023, focusing on GitLab's most prominent open-source projects. The selected projects follow agile software development methodologies and have task sizes recorded in Story Points. To extract data from GitLab, we developed a custom Python code that connects to the GitLab API for information retrieval. The code for this application is available on GitHub.

We collected only tasks where the *State* attribute was set to "Closed" and the *weight* attribute was populated. In GitLab, the *weight* field is used to record the effort in Story Points. Additional information about the projects included in the dataset can be accessed directly from GitLab.

The projects in the data set exhibit various characteristics that span various programming languages, business domains, and geographic locations of development teams. The primary entity in the data set is the User Story (or

	idproject	issuekey	created	title	description	storypoints
0	10152778	19541701	2019-03-28 15:04:16.070	(feat): change 'from' email addres…	Emails should not point to info@mi…	1.0
1	10152778	19273465	2019-03-20 02:11:19.496	(bug): Link previews disappear whe…	### Summary\r\n\r\nWhen editing a …	3.0
2	10152778	18774779	2019-03-04 11:21:14.993	(bug): subscribed blog post duplic…	NaN	2.0
3	10152778	18438523	2019-02-20 16:14:17.411	(bug): Cant delete a reply to a co…	### Summary\r\n\r\n0n ones own blo…	5.0
4	10152778	18177704	2019-02-12 16:55:06.846	Counter in group convo-feeds count…	### Summary\r\n\r\nWhen a comment …	5.0
5	10152778	18175263	2019-02-12 15:48:55.524	Subscribers list not complete, or …	### Summary\r\n\r\n0n some channel…	6.0
6	10152778	18102324	2019-02-10 03:50:42.211	(feat): admin tools for help desk	Add ability for admins to delete h…	10.0
7	10152778	18012969	2019-02-06 16:38:18.891	(feat): Convert ES reports data to…	See https://gitlab.com/minds/engin…	25.0
8	10152778	18010286	2019-02-06 15:17:53.730	(feat): Implement configuration ba…	A simple configuration update and …	5.0
9	10152778	18008315	2019-02-06 14:08:46.663	(feat): Implement user appeals in …	Dovetailing with our work to move …	15.0

Fig. 5 First columns of the data set

Table 3 Data set columns

Name	Description	Type
idproject	Key of the project	Categorical
issuekey	Key of the User Story	Categorical
created	Create data of the User Story	Date
title	User Story Title	Text
description	Full description of the User Story	ext
storypoint	Story point of the User Story	Numerical

Issue); other columns are the title, the description, and the created date from the User Story (Table 3).

The data set presented here includes projects that have not been used in previous studies. Although previous research has extracted data from the Jira management tool to build predictive models [5, 26, 29, 52], datasets sourced from GitLab projects are much rarer. Similarly to the approach taken by other researchers [5], we share all data collected during the study.

The expected contribution is that this dataset can assist in education and research on agile software development. Although our data set was initially designed for Story Points and User Story Estimation Training and Research, it also includes information relevant to other software engineering aspects. In addition, it provides the possibility to reproduce the findings from other studies.

It is worth noting that the NeoDataset has already been used by Zou et al. [53] and Moon et al. [54]. Ref. [53] used the NeoDataset as a basis to support and evaluate requirements elicitation texts with LLM assistance, contributing to the system's feasibility and validation. In turn, Ref. [54] used the NeoDataset in the development and validation of predictive models for effort estimation.

Hardware Requirements

The implementation of the proposed models requires consideration of both GPU capacity and system memory. For running large language models (LLMs) locally, the GPU is the most critical component, since the number of parameters directly affects video memory (VRAM) consumption. Models up to 7 billion parameters can be executed on mid-range GPUs with approximately 6–8 GB of VRAM, provided that quantization techniques (e.g., 4-bit) are applied.

In addition to the GPU, the system RAM also plays an important role, particularly in preprocessing tasks, vectorization (TF-IDF), and the execution of classical machine learning algorithms such as SVR. These modules do not require GPU acceleration, but they demand sufficient main memory to handle collections of thousands of user stories. A configuration between 16 and 32 GB of RAM is adequate for research and prototyping scenarios, while 64 GB or more is recommended in production environments or when processing long text contexts.

With regard to storage, sufficient disk space must be reserved to host language models in different formats. A 7B parameter model in FP16 precision occupies around 14 GB, while its 4-bit quantized version may reduce this to approximately 4–6 GB. Therefore, it is recommended to allocate at least 50 GB of free disk space, considering caches, intermediate data, and training artifacts. In this way, the hardware infrastructure should be planned according to the size of the selected model and the application's purpose, balancing cost, performance, and scalability.

It is worth noting that the experiments for this research were conducted on a modest personal computer equipped with an Intel Core i5 processor, 16 GB of RAM, and a GPU with 4 GB of VRAM. Although this configuration is below the recommended hardware for running larger LLMs in full precision, it was sufficient for smaller experiments by employing quantized models (4-bit) and limiting the context window and batch size. This highlights the feasibility of replicating similar experiments even in resource-constrained environments.

Screenshots

In the initial, the user inputs the text of the user story (Fig. 6). Subsequently, the system provides the corresponding recommendation (Fig. 7), the readability indices (Fig. 8), and the estimated Story Points (Fig. 9).

Results and Discussion

This section presents a quantitative and qualitative evaluation of the tool with the support of the TAM and Attrak-Diff frameworks and discusses the results. The survey was conducted in December 2023 with an online questionnaire in Google Forms. The completeness, quality, and suitability of the questionnaire was first examined by a panel of 6 experts with 7 years of experience in agile development.

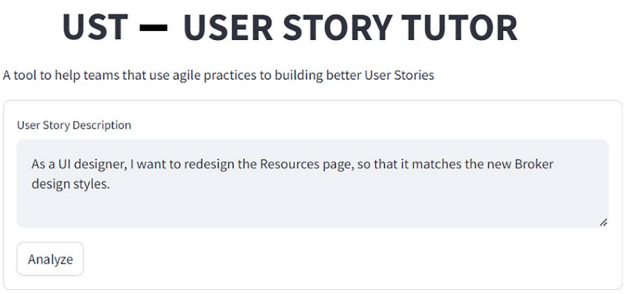


Fig. 6 Home screen

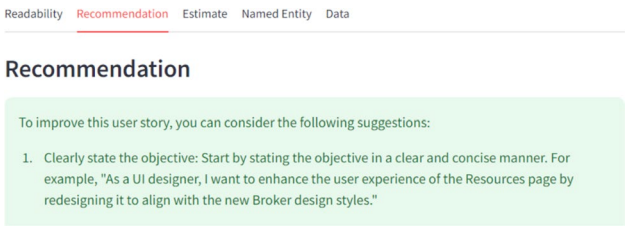


Fig. 7 Recommendation Module Screen

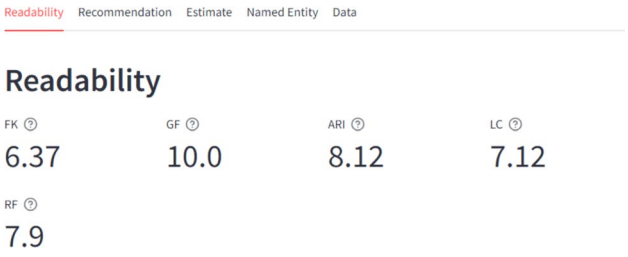


Fig. 8 Readability Indexes Module screen

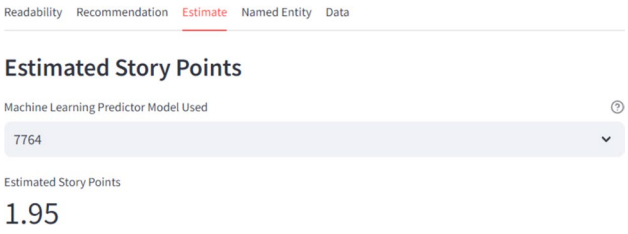


Fig. 9 Estimation screen

The questionnaire used a 5-level Likert scale to gauge the respondents' agreement with statements made concerning the proposal. The experts' comments and suggestions led to the adjustment of the questionnaire, which was then applied to a sample of respondents.

Our sample of survey respondents is made up of 40 participants. 70% of those who responded to the survey had worked directly with agile methodologies. More than half of these (56%) had worked in a software factory and had already worked as a member of the software development team. Of these, 20% have been Scrum Masters and 10% have been Product Owners. The other participants had generally participated in academic activities related to software engineering. On average, our sample consisted of professionals with 3 years of experience in agile methodology.

The four constructs of TAM are examined in detail below: perception of usability, perception of ease of use, external variables, and attitude.

Usability perception refers to the effectiveness with which users believe that the proposal improves their task performance. This aspect is critical because it determines the practical value that the tool adds to the user's workflow. To evaluate this, we analyzed the mean, median, and standard deviation of the responses of the participants on a Likert scale (Table 4). The threshold for a neutral response was set at "3" in our experiments, with values above this indicating a more positive perception of usability. A higher mean or median score would suggest that the participants felt that the proposal made a significant improvement in their efficiency and effectiveness of tasks, supporting the intended purpose of the tool [55].

The analysis of the responses revealed that most of the participants rated the proposal favorably in terms of usability, as evidenced by scores that exceeded the neutral threshold. This indicates that users generally found the proposal to be beneficial in helping them accomplish their tasks more efficiently (Table 4). This positive feedback suggests that the tool successfully meets its functional goals by improving the user's ability to create better user stories and estimate

Table 4 Technology assessment model statics

Construct	Question	Mean	Med	SD	Cronbach	IC
Perception of usability	Using the tool is useful to improve my User Stories	4.45	5.00	0.80	0.81	[0.64 0.90]
	I learned how to build better User Stories after using the tool	4.07	4.00	1.10		
Perceived ease of use	Learning to use the tool was easy for me	3.70	4.00	1.28	0.92	[0.87 0.95]
	Searching for information in this tool was simple	3.72	4.00	1.09		
	Accessing the tool is simple	4.07	4.00	1.14		
External variables	The application's navigation attributes - menu, icons, links, and buttons: are clear and easy to find	4.02	4.00	1.17	0.87	[0.77 0.93]
	The tool has a good interface	3.9	4.00	1.20		
Attitude	I believe it is better to use the tool to help create the user story than not to use it.	4.25	4.00	0.88	0.73	[0.49 0.85]
	I intend to use the tool to create better user stories and to plan my tasks better	3.85	4.00	1.15		

the effort with more precision. The favorable perception of usability aligns with the broader objectives of the Technology Assessment Model, which seeks to measure user satisfaction and productivity improvements as key outcomes.

The perception of ease of use refers to the extent to which users find the tool easy to learn and operate. This is a metric as it affects user adoption and overall satisfaction with the tool. In this evaluation, all average scores for ease of use exceeded the predetermined threshold, indicating a generally positive perception of the accessibility and user-friendliness of the tool. However, the standard deviation above 1 suggests a wider variation in user responses, meaning that while many users found the tool easy to use, there was some divergence in individual experiences, and some users could find it more challenging.

The analysis of factors of external variables that influence perceived utility and ease of use further enhances the understanding of how users interact with the tool. A median score above 4 suggests that users generally found external characteristics, such as interface design and navigation, well-executed and conducive to effective use of the tool. When examining the attitude construct, which reflects the intention of the users to continue using the tool, we also observe a mean score above the threshold, reinforcing the overall positive reception across all the measured constructs.

To ensure statistical reliability of the TAM findings, Cronbach's Alpha was used, following the method employed by [55, 56]. Cronbach's Alpha is a measure of internal consistency, ranging from 0 to 1, with higher values indicating a stronger reliability of the survey or questionnaire. In this investigation, a Cronbach Alpha value greater than 0.7 was set as the acceptable threshold, with a confidence level of 95%.

Based on the reported TAM values, almost all constructs met or exceeded the selected threshold, confirming that the internal consistency of the survey is robust and the results are statistically reliable. Consequently, we can conclude that the evaluation framework used in this study provides a good measure of user perceptions regarding the tool's usability and ease of use.

The portfolio of results from the AttrakDiff test [9] assesses key factors that provide a deeper understanding of the proposal, serving as a complement to the TAM framework. The short version of the AttrakDiff test(AttrakDiff Mini), comprising 10 questions, was used to collect user feedback and infer the metrics.

The AttrakDiff Mini consisting (Table 5) of 10 items organized into three main dimensions: Pragmatic Quality (PQ), which assesses functional aspects and usability; Hedonic Quality (HQ), which captures attributes related to identity, creativity, and stimulation; and Attractiveness (ATT), which summarizes the overall perception of the product's appeal. Each pair of opposite adjectives is evaluated on a 7-point scale, enabling the analysis of both practical utility and emotional factors in the user experience.

Table 5 AttrakDiff mini statements

Dimensions	Statements pair
Pragmatic Quality (PQ)	1. Complicated—Simple 2. Impractical—Practical 3. Unpredictable—Predictable 4. Confusing—Clearly structured
Hedonic Quality (HQ)	5. Tacky—Stylish 6. Cheap—Premium 7. Unimaginative—Creative 8. Dull—Captivating
Attractiveness (ATT)	9. Ugly—Attractive 10. Bad—Good

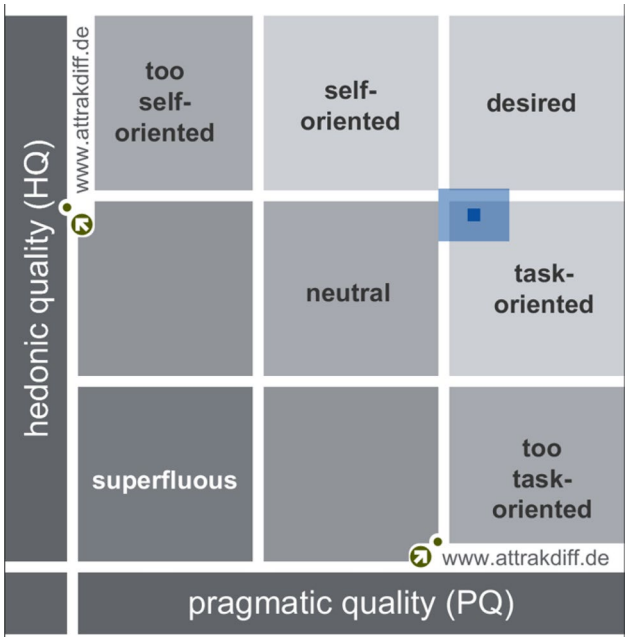


Fig. 10 Portfolio of results. Adapted from [45]

The vertical axis reflects the hedonic quality (with lower values at the bottom), while the horizontal axis represents the pragmatic quality (with lower values to the left) (see Fig. 10). The placement of the product within specific character regions is based on these values. A smaller confidence rectangle indicates a higher reliability of the results, reflecting the consistency in user evaluations. In contrast, a larger rectangle implies greater variability in the responses. In this case, the results point to a small confidence rectangle in the upper right quadrant, indicating strong task-oriented usability.

The hedonic quality, a central dimension in the AttrakDiff test, is divided into two main components: stimulation

and identity. As shown in the diagram showing the average values from the AttrakDiff test (Fig. 11), it provides a clear visualization of how the product scores on these dimensions. The stimulation component measures how engaging, exciting, and stimulating users find the product, highlighting its ability to evoke emotional responses. On the other hand, identity reflects the degree to which the product aligns with or enhances the user's self-image or brand identity, indicating how well users can relate to or see themselves in the product.

In addition, the diagram also includes an evaluation of the attractiveness of the product (ATT), which combines both hedonic and pragmatic qualities to assess the overall appeal of the product to its users. The attractiveness rating is particularly important because it represents a holistic view of how desirable the product is perceived. By examining these distinct dimensions together, the diagram provides a comprehensive understanding of user perception, focusing not only on practical usability but also on how emotionally appealing and stimulating the product is to its audience [9].

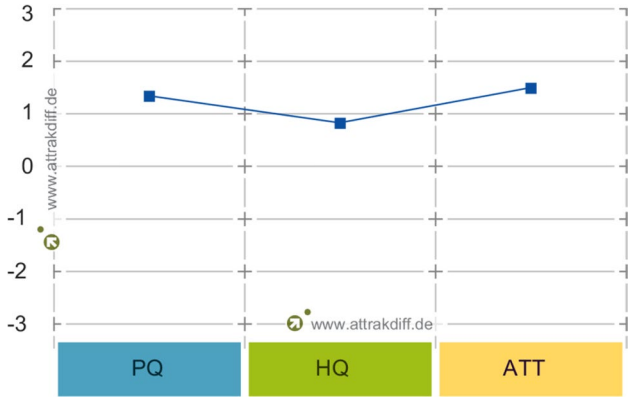


Fig. 11 Diagram of average values. Adapted from [45]

In Fig. 12, we provide a detailed description of the word pairs used in the AttrakDiff evaluation. The diagram presents the mean values of these word pairs, which are designed to assess various attributes of the product based on user perceptions. The word pairs represent opposing characteristics, and their placement on the diagram reflects the degree to which users associate the product with one characteristic over its opposite. Of particular interest are extreme values, as these highlight product attributes that are especially well resolved or are particularly problematic, offering key insights into its strengths and weaknesses [9].

In this evaluation, the better results, those in which users rated the product positively, are placed in the upper, more favorable quadrant. Based on the consolidated results shown in Fig. 12, the majority of the evaluated word pairs fell within the positive quadrant, indicating a generally favorable perception of the product. However, a notable exception was the pair "cheap-premium" In this case, the product was rated closer to the cheap side, suggesting that users did not perceive it as particularly premium. This isolated negative rating contrasts with the overall positive trend, highlighting a potential area for improvement.

Hyper-parameters: The SVR model was used in this study with its default hyperparameters, namely: `kernel='rbf', degree=3, gamma='scale', coef0=0.0, tol=0.001, C=1.0, epsilon=0.1, shrinking=True, cache_size=200, verbose=False`, and `max_iter=-1`. Likewise, the feature vector was generated using `TfidfVectorizer`, also with default parameters: `lowercase=True, analyzer='word', stop_words=None` (with stopwords removed during preprocessing), and `ngram_range=(1, 1)`. Although these settings yielded satisfactory results, further exploration of hyperparameter tuning strategies, such as grid search or randomized search, could

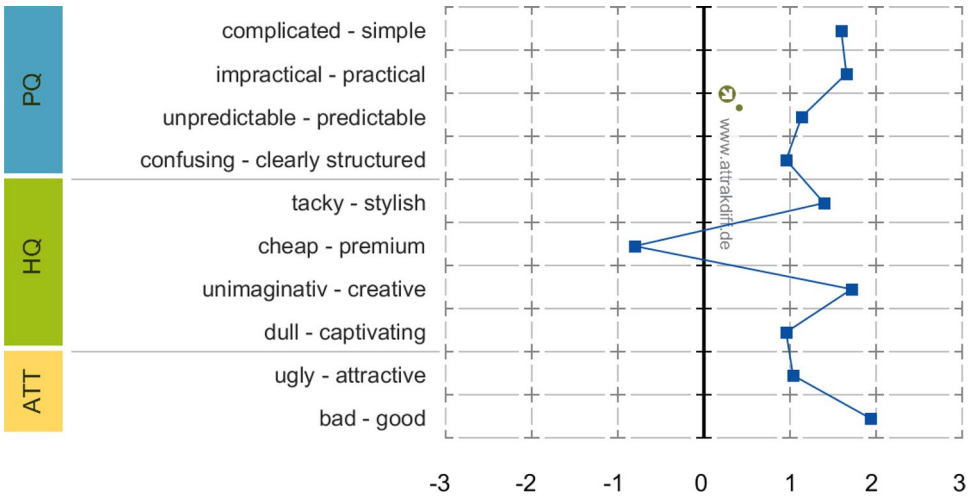


Fig. 12 Description of word-pairs. Adapted from [45]

enhance model performance and is recommended for future work.

Although the data set used in this study comprises 12,262 User Stories from 33 open-source GitLab projects, the scalability of the approach to larger and more heterogeneous data sets is a crucial aspect for broader adoption. The modular architecture of the tool, which features distinct components for recommendation, estimation, and readability, facilitates its extension to accommodate significantly larger datasets. Moreover, since the estimator module uses scalable machine learning techniques such as SVR with TF-IDF vectorization, it can be retrained with additional data from diverse business domains, improving generalizability.

We restrict the predictive algorithm to SVR for two main reasons: (i) SVR is the baseline for story-point prediction from user-story text used in [26], and it remains a strong benchmark to beat, even after later deep models were re-evaluated, which ensures a fair and comparable setup; ; and (ii) our per-project datasets are high-dimensional and relatively small, a regime where margin-based SVR generalizes well and is computationally efficient, enabling reproducible comparisons across projects. Finally, the use of Large Language Models (LLMs) for recommendations also enhances domain adaptability, as these models are pre-trained on extensive and varied corpora.

Security and privacy risks: Given that the tool interacts with external APIs such as OpenAI's GPT-3.5, we need to address potential data security and privacy risks, especially within corporate environments. When sensitive business information, such as proprietary User Stories or project descriptions, is transmitted to third-party servers for processing, there exists a risk of unintended data exposure, misuse, or interception. This concern becomes more pronounced in sectors that handle confidential data or operate under strict regulatory frameworks. Furthermore, the dependence of the tool on Internet connectivity and third-party infrastructure can introduce vulnerabilities related to data sovereignty and compliance with data protection laws. Therefore, organizations intending to adopt such tools must implement adequate safeguards, such as anonymization techniques, strict API usage policies, and explore the possibility of deploying on-premise or open source LLM alternatives to retain full control over data flow and confidentiality.

Data bias for non-software industries: Since the training data used in the estimator module originates predominantly from open-source software development projects hosted on GitLab, there is a risk that the learned patterns may reflect the specific practices, conventions, or linguistic styles of that domain. As a result, the performance of the model and the relevance of the recommendations can degrade when applied to nonsoftware industries, such as healthcare, finance, or education, where the structure and vocabulary of User Stories can differ significantly. Future work should consider incorporating data from a wider range of sectors to improve the model's generalization capabilities and to mitigate domain-specific biases, thus enhancing the versatility and fairness of the tool across diverse business environments.

Open-weight: Although the current proposal uses OpenAI's GPT-3.5-turbo, several open-weight language models offer comparable performance and can serve as viable alternatives, especially in scenarios requiring infrastructure control or enhanced data privacy. Models such as Gemma 7B Instruct (Google) and Mistral 7B Instruct have demonstrated strong instruction-following capabilities, delivering results on par with GPT-3.5 in tasks such as text generation and rewriting. Another promising option is LLaMA 3 8B (Meta), which provides robust performance, particularly in reasoning and natural language understanding tasks. For environments with greater computational resources, Mixtral 8x7B, based on a mixture-of-experts architecture, also proves to be an effective alternative. All of these models are released under open or permissive licenses and can be integrated into the proposed architecture as replacements for GPT-3.5, ensuring high-quality recommendations while allowing more flexible adoption in corporate or academic settings with connectivity or compliance constraints.

Token cost: The cost of 1 million tokens using GPT-3.5 Turbo is approximately $0.50 for the input tokens and $1.50 for the output tokens, totaling up to $2.00. Based on our implementation, each recommendation request involved approximately 300 tokens in the input (including the user story and a hidden prompt) and generated around 150 tokens in the output, totaling an estimated 450 tokens per request. With 40 participants, even under the assumption that each user submitted up to three user stories, the total usage would be approximately 54,000 tokens. Given the average price of GPT-3.5 at $0.0035 per 1,000 tokens, the total cost would remain well below one dollar. Although this information may be less relevant today due to the rapid decrease in LLM costs, it can still offer valuable information to institutions and companies considering the adoption of LLMs to improve user stories.

Diverse textual representation strategies: Although our system employs distinct text representation strategies across its modules, textual embeddings from a neural language model (GPT-3.5) for the Recommender and static representations based on TF-IDF for the Estimator, this duality was a deliberate design choice aligned with the specific demands of each task. Contextual representations are well-suited for generating nuanced recommendations, as they capture deep semantic relationships. In contrast, the effort estimation task, treated as a regression problem,

benefits from the interpretability, simplicity, and proven effectiveness of TF-IDF vectors when combined with SVR, as supported by a study in this domain [26]. However, we acknowledge that integrating contextual embeddings, such as sentence transformer or BERT-derived vectors, into the estimator could offer richer semantic representations and is a promising avenue for future research. This modular approach also allows for easy experimentation with more sophisticated embeddings, ensuring that the architecture remains extensible as NLP advances.

Related Work

Improving the quality of User Stories is an emerging area of research driven by advancements in artificial intelligence. Traditionally, the most common approaches involve transforming the User Story into an intermediate model, such as a use case diagram [57], or employing other natural language processing techniques to generate interpretable reports (e.g., AQUSA) [58].

However, using an intermediate model adds complexity to the solution, which may be seen as an undesirable feature. Our approach avoids this by not relying on an intermediate model. Instead, we used text readability indexes, well established in fields such as economics and literature, for text analysis, combined with OpenAI's LLM to provide personalized recommendations. In addition, the estimator module utilizes machine learning and natural language processing techniques.

Compared to previous tools, such as AQUSA [58], which focus on detecting missing information without providing tailored recommendations, our proposal adds value by actively guiding the user toward improvements.

A related work worth noting is the USQA tool [17], which also leverages natural language processing to enhance User Story quality by checking for completeness, usefulness, and ambiguity. However, USQA lacks the capability to provide personalized recommendations or effort estimates such as our proposal. Table 6 compares UST with the AQUSA and USQA proposals and illustrates UST's contribution compared to that of existing related work to User Story writing.

Additionally, while both tools focus on improving User Stories, our proposal incorporates a more comprehensive evaluation framework using both TAM and AttrakDiff, providing robust feedback on its usability, practicality, and overall quality. This emphasis on user feedback, combined with the tool's practical focus on readability and effort estimation, sets the proposal apart from its predecessors by offering a more user-centered and functionality-rich solution.

Table 6 Comparison with related work

Tool	Recommend report	User Stories validated	Use LLM	Use readability index
UST (this)	Yes	40	Yes	Yes
AQUSA	Yes	1.023	No	No
USQA	Yes	35	No	No

The task description contains relevant information for automatic estimation, and the SVR was the one that obtained the best results in the story point predictor [26]. In addition, it is known that more structured texts allowed for a more efficient prediction [59] and SVR and Linear Regression were the ones that showed the best results [52].

However, it is important to note that direct comparison between the effort estimation predictors presented in this and related studies should be approached with caution, as they are trained and evaluated on different datasets. The dataset used in this work was built from open-source projects, which differ in domain, language, and team dynamics from those used in other studies. Consequently, variations in model performance may be as much a reflection of the data sources as of the modeling techniques themselves. Nevertheless, to the best of our efforts, the predictor developed in this study employs techniques that are most recommended in the related literature, such as the use of TF-IDF vectorization combined with SVR. This choice reflects both empirical effectiveness and methodological alignment with prior work in the area.

Limitations and Threats

Readability indices should be interpreted with caution, as their formulas rely solely on a few variables, such as complex words and sentence length. Consequently, they are unable to assess the cohesion and coherence of a business User Story, which involves semantic, syntactic, and pragmatic factors. In addition, readability, in this context, goes beyond syntactic complexity and should consider the domain-specific semantics, structure, and clarity required in agile practices. One promising direction involves leveraging Natural Language Processing (NLP) techniques, particularly the use of Large Language Models (LLMs), which are capable of interpreting context, detecting ambiguity, and evaluating coherence beyond surface-level metrics. Integrating LLM-based assessments for readability purposes could offer a more nuanced understanding of textual clarity tailored to software artifacts, supplementing traditional indexes with semantic-based insights. This hybrid approach would

enhance the evaluation of User Stories' readability, providing more meaningful feedback to agile teams.

Story point estimation typically follows the Fibonacci scale. However, in our proposal, the estimator returns a real number between 0 and 100, treating the task as a regression problem rather than a classification one. Although this approach provides flexibility, the use of the Fibonacci scale could offer greater interpretability of the results.

In addition to the limitations already discussed, one potential threat is reliance on the specific data set used to train the effort estimation model. The dataset, which was sourced from open source projects, may not fully represent the diversity of software development practices across different industries, teams, or project scales. This could lead to biased predictions in environments that differ significantly from the projects used to train the model. Furthermore, the generalization of the proposed recommendations and story point estimates could be limited in specialized domains where User Stories have unique requirements or follow non-standard formats.

Another limitation is that the proposal can struggle with ambiguous, overly complex, or context-sensitive language, leading to incorrect recommendations or effort estimations. Furthermore, the integration of large language models like OpenAI models introduces concerns regarding data privacy, since these models require Internet access and could expose sensitive project information. Ensuring data security and addressing privacy concerns in enterprise environments could present a major challenge for the proposal's widespread adoption.

The use of an LLM model provided by companies via API (e.g., OpenAI's ChatGPT) links the solution to a specific corporate entity with a financial cost.

Finally. There is some criticism in the literature on the numerical interpretation of Likert scale questionnaires, particularly in the calculation of mean or averages [60]. To address this concern, we incorporated an additional framework for analyzing software quality, the AttrakDiff.

Conclusion

In this experiment, we evaluated a tool designed to recommend best practices for writing User Stories using a Large Language Model, alongside a User Story estimation module powered by Machine Learning, and a readability index presentation for User Story descriptions. The tool was assessed by software engineering practitioners using the TAM and AttrakDiff frameworks. The results indicate that the proposed tool effectively meets its objectives and has been well received by its target users.

In comparison to previous tools, the proposal goes beyond merely identifying incomplete or ambiguous User Stories by offering actionable feedback and integrating effort estimation based on historical data. This holistic approach is particularly useful for agile teams seeking to improve planning accuracy and story quality. In addition, this proposal added capabilities, such as recommending improvements and calculating effort estimates, to make it a more practical and advanced solution.

Based on this investigation, it can be concluded that a tool to assist in the construction of User Stories is a viable approach which can at the very least be used to educate teams on how to write better User Stories. The results of the evaluation experiment suggest that the proposed tool can effectively help agile practitioners by providing valuable feedback to improve their User Stories.

We also introduced a data set that contains data from GitLab-mined projects, which was used to train the predictive model for Story Points estimation. This data set is available for use in other research related to agile software development.

Future independent work could involve further validation experiments, including integration and evaluation within computer-based education platforms for agile software development methods. Another suggestion is to explore other natural language techniques, for example named entity recognition to extract relevant entities from the User Story text. Finally, we recommend the use with LLM models such as Gemma 3 4B due to its balance between performance and computational cost, as well as extensive documentation and community support.

Funding This work was carried out with the support of the Coordination for the Improvement of Higher Education Personnel - Brazil (CAPES) - Financing Code 001.

Data Availability The data set is available on Mendeley Data https://data.mendeley.com/datasets/skk2wn9j86 and Hugging Face https://huggingface.co/spaces/giseldo/userstory. The app can be tested at https://userstoryteach.streamlit.app and the source code at https://github.com/giseldo/userstory and https://github.com/giseldo/neo-gitlab-extractor.

Declarations

Conflict of Interest On behalf of all authors, the corresponding author states that there is no conflict of interest.

References

1. StandishGroup: The CHAOS report. http://www.standishgroup.com Accessed 2023 2015.
2. Sommerville I. Software engineering, 9th edn. 2011.
3. Cohn M. User Stories applied: for agile software development. Boston: Pearson Education; 2004.

4. Jadhav D, Kundale J, Bhagwat S, Joshi J. A systematic review of the tools and techniques in distributed agile software development. In: Agile software development: trends, challenges and applications, 2023;pp. 161–186.
5. Tawosi V, Al-Subaihin A, Moussa R, Sarro F. A versatile dataset of agile open source software projects. In: Proceedings—2022 Mining Software Repositories Conference, MSR 2022, vol. 1, pp. 707–711. Association for Computing Machinery, USA 2022. https://doi.org/10.1145/3524842.3528029.
6. Andrade AFM. Uma Abordagem Baseada Em Gamificação Para Estimativa De Esforço Em Desenvolvimento Ágil De Software. PhD thesis, Universidade Federal de Campina Grande 2021.
7. Wieringa RJ. Design science methodology for information systems and software engineering. Berlin: Springer; 2014.
8. Davis FD, Bagozzi RP, Warshaw PR. User acceptance of computer technology: a comparison of two theoretical models. Manag Sci. 1989;35(8):982–1003.
9. Hassenzahl M, Burmester M, Koller F. Attrakdiff: Ein fragebogen zur messung wahrgenommener hedonischer und pragmatischer qualität. In: Mensch & Computer 2003: Interaktion in Bewegung, 2003;pp. 187–196.
10. Sutherland J. Scrum: the art of doing twice the work in half the time. New York: Random House; 2014.
11. Dybå T, Dingsøyr T. Empirical studies of agile software development: a systematic review. Inf Softw Technol. 2008;50(9–10):833–59. https://doi.org/10.1016/j.infsof.2008.01.006.
12. PMI: Success Rates Rise - 2017 9th Global Project Management Survey. Technical report, PMI 2017. https://www.pmi.org/-/media/pmi/documents/public/pdf/learning/thought-leadership/pulse/pulse-of-the-profession-2017.pdf.
13. Trimble J, Shirley MH, Hobart SG. Agile: from software to mission system. In: 14th International Conference on Space Operations, 2016;2016:1–8. https://doi.org/10.2514/6.2016-2477.
14. Rigby DK, Sutherland J, Noble A. Agile scale: how to go from teams to hundreds. Hav Bus Rev. 2018;1–3.
15. Mergel I. Agile innovation management in government: a research agenda. Gov Inf Q. 2016;33(3):516–23. https://doi.org/10.1016/j.giq.2016.07.004.
16. Lucassen G, Dalpiaz F, Van Der Werf JME, Brinkkemper S. Forging high-quality user stories: towards a discipline for agile requirements. In: 2015 IEEE 23rd International Requirements Engineering Conference (RE), 2015;126–135. IEEE.
17. Jiménez S, Alanis A, Beltrán C, Juárez-Ramírez R, Ramírez-Noriega A, Tona C. Usqa: a user story quality analyzer prototype for supporting software engineering students. Comput Appl Eng Educ. 2023.
18. Choudhury P, Crowston K, Dahlander L, Minervini MS, Raghuram S. GitLab: work where you want, when you want. J Organ Des. 2020. https://doi.org/10.1186/s41469-020-00087-8.
19. Dimitrijević S, Jovanović J, Devedžić V. A comparative study of software tools for user story management. Inf Softw Technol. 2015;57:352–68.
20. Mani S, Sankaran A, Aralikatte R. Deeptriage: exploring the effectiveness of deep learning for bug triaging. In: ACM International Conference Proceeding Series, 2019;171–179. https://doi.org/10.1145/3297001.3297023. arXiv:1801.01275.
21. Chaparro O, Lu J, Zampetti F, Moreno L, Di Penta M, Marcus A, Bavota G, Ng V. Detecting missing information in bug descriptions. In: Proceedings of the ACM SIGSOFT Symposium on the Foundations of Software Engineering Part 2017;F130154:396–407. https://doi.org/10.1145/3106237.3106285.
22. Choetkiertikul M, Dam HK, Tran T, Ghose A, Grundy J. Predicting delivery capability in iterative software development. IEEE Trans Softw Eng. 2018;44(6):551–73. https://doi.org/10.1109/TSE.2017.2693989.
23. Ortu M, Destefanis G, Adams B, Murgia A, Marchesi M, Tonelli R. The JIRA repository dataset: Understanding social aspects of software development. In: ACM International Conference Proceeding Series **2015-October** 2015. https://doi.org/10.1145/2810146.2810147.
24. Ortu M, Murgia A, Destefanis G, Tourani P, Tonelli R, Marchesi M, Adams B. The emotional side of software developers in JIRA. In: Proceedings—13th Working Conference on Mining Software Repositories, MSR 2016;2016:480–483. https://doi.org/10.1145/2901739.2903505.
25. Valdez A, Oktaba H, Gomez H, Vizcaino A. Sentiment analysis in jira software repositories. In: Proceedings—2020 8th Edition of the International Conference in Software Engineering Research and Innovation, CONISOFT 2020:2020;254–259. https://doi.org/10.1109/CONISOFT50191.2020.00043.
26. Porru S, Murgia A, Demeyer S, Marchesi M, Tonelli R. Estimating story points from issue reports. In: ACM International Conference proceeding series. 2016. https://doi.org/10.1145/2972958.2972959.
27. Soares RGF. Effort Estimation via Text Classification And Autoencoders. In: 2018 International Joint Conference on Neural Networks (IJCNN), vol. 2018-July, pp. 1–8. IEEE, Rio de Janeiro, Brazil 2018. https://doi.org/10.1109/IJCNN.2018.8489030.
28. Dragicevic S, Celar S, Turic M. Bayesian network model for task effort estimation in agile software development. J Syst Softw. 2017;127:109–19. https://doi.org/10.1016/j.jss.2017.01.027.
29. Choetkiertikul M, Dam HK, Tran T, Pham T, Ghose A, Menzies T. A deep learning model for estimating story points. IEEE Trans Softw Eng. 2019;45(7):637–56. https://doi.org/10.1109/TSE.2018.2792473. arXiv:1609.00489.
30. Tawosi V, Moussa R, Sarro F. Deep learning for agile effort estimation have we solved the problem yet?, 2022;1–17.
31. Gavidia-Calderon C, Sarro F, Harman M, Barr ET. The Assessor's dilemma: improving bug repair via empirical game theory. IEEE Trans Softw Eng. 2021;47(10):2143–61. https://doi.org/10.1109/TSE.2019.2944608.
32. Huang Y, Wang J, Wang S, Liu Z, Wang D, Wang Q. Characterizing and predicting good first issues. In: International Symposium on empirical software engineering and measurement. 2021. https://doi.org/10.1145/3475716.3475789.
33. Umer Q, Liu H, Illahi I. CNN-based automatic prioritization of bug reports. IEEE Trans Reliab. 2020;69(4):1341–54. https://doi.org/10.1109/TR.2019.2959624.
34. Ricci F, Rokach L, Shapira B. Introduction to recommender systems handbook. In: Recommender systems handbook. Boston: Springer; 2010. p. 1–35.
35. Odilinye L, Popowich F. Personalized recommender system using learners' metacognitive reading activities. In: Methodologies and Intelligent Systems for Technology Enhanced Learning, 10th International Conference, 2020;195–205. Springer.
36. Bobadilla J, Ortega F, Hernando A, Gutiérrez A. Recommender systems survey. Knowl-Based Syst. 2013;46:109–32.
37. Burke R. Hybrid web recommender systems. In: The adaptive web: methods and strategies of web personalization, 2007;377–408.
38. Molla YS, Alemneh E, Yimer ST. COSMIC-based early software size estimation using deep learning and domain-specific BERT. IEEE Access. 2025;13(February):28463–75. https://doi.org/10.1109/ACCESS.2025.3540548.
39. OpenAI: GPT-4 Technical Report 2023;4:1–100. arXiv:2303.08774.
40. Garg S, Sharma DK. Linguistic features based framework for automatic fake news detection. Comput Ind Eng. 2022;172:108432. https://doi.org/10.1016/j.cie.2022.108432.

41. DuBay WH. The principles of readability: a brief introduction to readability research. Impact Inform. 2004;949:1–72.
42. Koenke K. Another practical note on readability formulas. J Read. 1971;15(3):203–8.
43. Bogert J. In defense of the fog index. Bull Assoc Bus Commun. 1985;48(2):9–12.
44. Gross PP, Sadowski K. FOGINDEX: a readability formula program for microcomputers. J Read. 1985;28(7):614–8.
45. Neo, G, Moura J, Almeida H, Neo A, Freitas Júnior, O. User Story Tutor (UST) to support agile software developers. In: Proceedings of the 16th International Conference on Computer Supported Education - Volume 2: CSEDU, pp. 51–62. SciTePress, Setúbal, Portugal 2024. https://doi.org/10.5220/0012619200003693 . INSTICC.
46. Textstat: textstat/textstat: python package to calculate readability statistics of a text object - paragraphs, sentences, articles. 2023. https://github.com/textstat/textstat. Accessed 2023.
47. Rossum G, Drake FL. Python reference manual. Amsterdam: Centrum voor Wiskunde en Informatica Amsterdam; 1995.
48. Neo B, Martin A. Streamlit—a faster way to build and share data apps. 2024. https://streamlit.io/. Accessed 26 Sep 2024.
49. Pedregosa F, Varoquaux G, Gramfort A, Michel V, Thirion B, Grisel O, et al. Scikit-learn: machine learning in python. J Mach Learn Res. 2011;12(Ot):2825–30.
50. Torrent TT, Hoffmann T, Almeida AL, Turner M. Copilots for linguists: Ai, constructions, and frames. In: Elements in construction grammar. 2023.
51. Enevoldsen K. A Python library for calculating a large variety of metrics from text(s) using spaCy v.3 pipeline components and extensions. 2024. https://pypi.org/project/textdescriptives/. Accessed 26 Sep 2024.
52. Scott E, Pfahl D. Using developers' features to estimate story points. In: ACM International Conference Proceeding Series. 2018;106:106–10. https://doi.org/10.1145/3202710.3203160.
53. Zou X, Liu Y, Shi X, Yang C. Goal2Story: a multi-agent fleet based on privately enabled sLLMs for impacting mapping on requirements elicitation vol. 1. Association for Computing Machinery, ??? 2025. http://arxiv.org/abs/2503.13279.
54. Moon K, Jeon E, Park D, Jungwon Seo BP, Kim S. A study on improving story point data prediction performance using a combination of tf-idf and sbert text features. In: Proceedings of KIIT Conference, 2025;417–422.
55. Dantas E, Costa AAM, Vinicius M, Perkusich MB, Almeida HO, Perkusich A. An effort estimation support tool for agile software development: an empirical evaluation. In: SEKE, 2019;82–116.
56. Gliem JA, Gliem RR. Calculating, interpreting, and reporting Cronbach's alpha reliability coefficient for likert-type scales. In: Midwest Research-to-Practice Conference in adult, continuing, and community education. 2003.
57. Elallaoui M, Nafil K, Touahni R. Automatic transformation of user stories into uml use case diagrams using nlp techniques. Proc Comput Sci. 2018;130:42–9.
58. Lucassen G, Dalpiaz F, Werf JME, Brinkkemper S. Improving agile requirements: the quality user story framework and tool. Requir Eng. 2016;21:383–403.
59. Abrahamsson P, Fronza I, Moser R, Vlasenko J, Pedrycz W. Predicting development effort from user stories. In: International Symposium on Empirical Software Engineering and Measurement, 2011;400–403. https://doi.org/10.1109/esem.2011.58.
60. Fávero LP, Belfiore P. Manual de Análise de Dados: Estatística e Modelagem Multivariada Com Excel®, SPSS® e Stata®. Rio de Janeiro: Elsevier Brasil; 2017.

Apêndice I

Código-Fonte

Prompt I.1: Criação do modelo Cross-Project

```
import pandas as pd
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.svm import SVR
import re
import spacy
import joblib
import os
import pandas as pd

nlp = spacy.load("en_core_web_sm", disable=["ner", "parser"])
stopwords = nlp.Defaults.stop_words

def preprocess_text(title, description):
    nlp = spacy.load("en_core_web_sm")
    nlp.select_pipes(disable=["ner", "parser"])
    stopwords = nlp.Defaults.stop_words
    contexto = title + ". " + description
    contexto = re.sub("www\S+|http\S+|@\S+|\s+", " ", contexto)
    def tokens(text):
        doc = nlp(text)
        tokens = [token.lemma_ for token in doc if token.is_alpha and
            token.text not in stopwords]
        return ' '.join(tokens)
    contexto = tokens(contexto)
```

```
24      contexto = contexto.lower()
25      return contexto
26
27  def generate_model():
28          df = pd.read_csv("hf://datasets/giseldo/neodataset/issues.csv")
29          df.dropna(inplace=True)
30          df = df[df.storypoints.between(df.storypoints.quantile(.05), df.
                storypoints.quantile(.95))] # remove outliers based on the 5th
                 and 95th percentiles
31          df["context"] = df["title"] + ". " + df["description"]
32          df["context"] = df["context"].apply(preprocess_text)
33          X = df["context"]
34          y = df["storypoints"]
35          vec = TfidfVectorizer()
36          X_bow_matrix = vec.fit_transform(X)
37          reg = SVR()
38          reg.fit(X_bow_matrix, y)
39          os.makedirs('models', exist_ok=True)
40          joblib.dump(vec, os.path.join('models', f'cross_project.vec'))
41          joblib.dump(reg, os.path.join('models', f'cross_project.model'))
42
43  if __name__ == "__main__":
44          generate_model()
```

Prompt I.2: Código Fonte do User Story Tutor

```
1  import streamlit as st
2  import textdescriptives as td
3  import spacy
4  from spacy import displacy
5  import numpy as np
6  from openai import OpenAI, AuthenticationError
7  import joblib
8  from dotenv import load_dotenv
9  import re
10
11 load_dotenv()
12
```

```
13 API_KEY = st.sidebar.text_input("OPENAI KEY", type="password")
14 if not API_KEY:
15     st.sidebar.error("Please enter a valid API key to continue.")
16
17 nlp =spacy.load("en_core_web_sm")
18
19 col1, col2, col3, col4, col5 = st.columns(5)
20 with col1:
21     usereadability = st.checkbox("Use Readability Indexes", False)
22 with col2:
23     usellm = st.checkbox("Use LLM recommendation", False)
24 with col3:
25     useEstimator = st.checkbox("Use Estimator Cross-project", False)
26 with col4:
27     useNER = st.checkbox("Use NER", False)
28 with col5:
29     useBasicTextFeatures = st.checkbox("Use Basic Text Features", False)
30
31 with st.form(key="frm_principal"):
32     txttitle = st.text_input(label="User Story Title", value="Redesign a
          resource page")
33     txtuser = st.text_area(label="User Story Description", value="As a UI
           designer, I want to redesign the Resources page, so that it
          matches the new Broker design styles.")
34     btn_submit = st.form_submit_button(label="Analyze")
35
36 def get_nearest_fibonacci(value):
37     """Retorna o número de Fibonacci mais próximo do valor fornecido"""
38     fibonacci_sequence = [1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, 233]
39
40     if value <= 1:
41         return 1
42
43     closest = fibonacci_sequence[0]
44     min_diff = abs(value - closest)
45
46     for fib in fibonacci_sequence:
```

```
47          diff = abs(value - fib)
48          if diff < min_diff:
49              min_diff = diff
50              closest = fib
51
52      return closest
53
54  def preprocess_text(title, description):
55      nlp = spacy.load("en_core_web_sm")
56      nlp.select_pipes(disable=["ner", "parser"])
57      stopwords = nlp.Defaults.stop_words
58      text = title + ". " + description
59      text = re.sub(r"www\S+|http\S+|@\S+|\s+", " ", text)
60      doc = nlp(text)
61      tokens = [token.lemma_.lower() for token in doc if token.is_alpha and
            token.text.lower() not in stopwords]
62      return ' '.join(tokens)
63
64  stLeg, stR, stEE, stNER, stD = st.tabs(["Readability Indexes", "LLM
      recommendation", "Estimator Cross-project", "NER", "Basic Text
      Features" ])
65
66  if btn_submit:
67      with stNER:
68          if useNER:
69              st.header("Named Entity Recognition - NER" )
70              doc = nlp(txtuser)
71              dep_svg = displacy.render (doc, style="dep", jupyter=False)
72              st.subheader("Entity visualizer")
73              ent_html = displacy.render(doc, style="ent", jupyter=False)
74              st.markdown(ent_html, unsafe_allow_html=True)
75
76      with stLeg:
77          if usereadability:
78              df = td.extract_metrics(text=txtuser, spacy_model="
                    en_core_web_sm", metrics=None)
79              st.header("Readability" )
```

```
80              sst1, sst2, sst3, sst4 = st.columns(4)
81
82              FK = value=df["flesch_kincaid_grade"]
83              GF = value=df["gunning_fog"]
84              ARI = df["automated_readability_index"]
85              LC = value=df["coleman_liau_index"]
86              RF = round(np.mean([FK, GF, ARI, LC]), 2)
87
88              sst1.metric(label="FK", value=round(FK,2), help= "Teste de
                    facilidade de leitura de Flesch")
89              sst2.metric(label="GF", value=round(GF,2), help="Gunning fog
                    index")
90              sst3.metric(label="ARI", value=round(ARI,2), help="Automated
                    Readability Index - ARI")
91              sst4.metric(label="LC", value=round(LC,2), help="Coleman-Liau
                     index")
92              ssst1, ssst2, ssst3, ssst4 = st.columns(4)
93              ssst1.metric(label="RF", value= RF, help="Resultado Final. Mé
                    dia aritmética entre os indicadores FK, GF, ARI e LC")
94
95      with stEE:
96          if useEstimator:
97              st.header("Story Points Estimator")
98
99              contexto = preprocess_text(txttitle, txtuser)
100             model = joblib.load(f"models/cross_project.model")
101             vec = joblib.load(f"models/cross_project.vec")
102             X_bow_matrix = vec.transform([contexto])
103             sp = model.predict(X_bow_matrix)
104             fibonacci_sp = get_nearest_fibonacci(sp[0])
105             st.metric(label="Estimated Story Points", value=f"{
                    fibonacci_sp} (original: {round(sp[0],2)})")
106
107     with stR:
108         if usellm:
109             st.subheader("LLM Recommendation")
110             try:
```

```
111                 client = OpenAI(api_key=API_KEY)
112                 completion = client.chat.completions.create(
113                 model="gpt-3.5-turbo",
114                 messages=[{"role": "system", "content": "You are a scrum
                        master, skilled in create better user story for agile
                        software projects."},
115                  {"role": "user", "content":"How can i improve this
                            this user story : {}".format(txtuser)}]
116                 )
117                 st.success(completion.choices[0].message.content)
118             except AuthenticationError as e:
119                 st.error("Authentication error: Invalid or missing API
                        key. Please check your OpenAI key.")
120             except Exception as e:
121                 st.error("Unexpected error communicating with OpenAI API
                        .")
122
123             st.warning("This module provides recommendations to improve
                    the writing of your User Story.")
124
125     with stD:
126         if useBasicTextFeatures:
127             df = td.extract_metrics(text=txtuser, spacy_model="
                    en_core_web_sm", metrics=None)
128             st.header("Basic Text Features" )
129             st.dataframe(df.T)
```

Prompt I.3: Modelo Zero-Shot

```
1  import ollama
2  import pandas as pd
3  from typing import Dict, List
4  from uuid import uuid4
5
6  class StoryPointsEstimationModel:
7      def __init__(self, model_name: str = "gemma3"):
8          """Inicializa o modelo com o nome do modelo Ollama."""
9          self.model_name = model_name
```

```
10
11      def preprocess_data(self, user_story_data: Dict) -> str:
12          """Converte os dados da user story em um prompt estruturado para
                o LLM."""
13          prompt = (
14              "Você é um especialista em estimativa de esforço para
                    projetos ágeis. "
15              "Com base no texto da user story e sua descrição, estime os
                    story points necessários para completar a tarefa. "
16              "Considere fatores como complexidade, volume de trabalho e
                    riscos implícitos. "
17              "Retorne apenas um número inteiro representando os story
                    points (ex.: 1, 2, 3, 5, 8, 13, etc.).\n\n"
18              f"Texto da user story: {user_story_data['user_story_text']}\n
                    "
19              f"Descrição da user story: {user_story_data['description']}\n
                    "
20              f"Retorne sempre apenas um número inteiro representando os
                    story points (ex.: 1, 2, 3, 5, 8, 13, etc.).\n\n"
21              f"Não retorne nenhum texto além do número inteiro.\n\n"
22              f"Não retorne nenhum texto além do número inteiro.\n\n"
23          )
24          return prompt
25
26      def call_llm(self, prompt: str) -> int:
27          """Faz a chamada ao modelo Ollama e retorna a estimativa de story
                points."""
28          try:
29              response = ollama.generate(
30                  model=self.model_name,
31                  prompt=prompt,
32                  options={
33                      "temperature": 0.3,
34                      "num_predict": 10
35                  }
36              )
37              return int(response.get("response", "0").strip())
```

```
38          except ValueError as e:
39              print(f"Erro ao converter a resposta do Ollama para número
                    inteiro: {str(e)}")
40              # raise Exception(f"Erro ao converter a resposta do Ollama
                    para número inteiro: {str(e)}")
41              return int(-1)
42          except Exception as e:
43              raise Exception(f"Erro ao chamar o modelo Ollama: {str(e)}")
44
45      def estimate_story_points(self, user_story_data: Dict) -> int:
46          """Estima os story points para uma user story usando o LLM."""
47          prompt = self.preprocess_data(user_story_data)
48          story_points = self.call_llm(prompt)
49          return story_points
50
51      def batch_estimate(self, user_stories: List[Dict]) -> pd.DataFrame:
52          """Estima os story points para várias user stories e retorna um
                DataFrame com os resultados."""
53          results = []
54          for story in user_stories:
55              story_points = self.estimate_story_points(story)
56              results.append({
57                  "story_id": story.get("story_id", str(uuid4())),
58                  "estimated_story_points": story_points,
59                  "real_story_points": story.get("real_story_points", None)
60              })
61          return pd.DataFrame(results)
62
63  def generate_llm_zero_shot_model(project_name):
64      model = StoryPointsEstimationModel()
65
66      df = pd.read_csv("deep-se.csv")
67      df_jirasoftware = df[df["project"] == project_name]
68
69      df_jirasoftware = df_jirasoftware[df_jirasoftware['storypoint'] != 0]
70      df_jirasoftware = df_jirasoftware.dropna(subset=['storypoint', 'title
            ', 'description'])
```

```
71
72      # Calculate the split index for 70-30 split
73      split_idx = int(len(df_jirasoftware) * 0.7)
74
75      # Split the data into train and test sets
76      df_train = df_jirasoftware.iloc[:split_idx]
77      df_test = df_jirasoftware.iloc[split_idx:]
78
79      print(f"Training set size: {len(df_train)}")
80      print(f"Test set size: {len(df_test)}")
81
82      user_stories = df_test.apply(lambda row: {
83          "story_id": str(row.get("issuekey", uuid4())),
84          "user_story_text": row.get("title", ""),
85          "description": row.get("description", ""),
86          "real_story_points": row.get("storypoint", None)
87      }, axis=1).tolist()
88
89      result_df = model.batch_estimate(user_stories)
90
91      result_df.to_csv("result_llm_zero_shot_{}.csv".format(project_name),
            index=False)
92
93  if __name__ == "__main__":
94      project_name = "jirasoftware"  # exemplo project name
95      generate_llm_zero_shot_model(project_name)
```

Prompt I.4: Criação do modelo LLM

```
1  from transformers import AutoTokenizer,
       AutoModelForSequenceClassification, Trainer, TrainingArguments
2  from datasets import Dataset
3  from sklearn.model_selection import train_test_split
4  import pandas as pd
5  import re
6
7  def remove_html_tags(text):
8      if isinstance(text, str):
```

```
9           clean_text = re.sub(r'<[^>]+>', '', text)
10          clean_text = re.sub(r'^{html}', '', clean_text)
11          return clean_text.strip()
12      return text
13
14  def remove_urls(text):
15      if isinstance(text, str):
16          url_pattern = r'http[s]?://(?:[a-zA-Z]|[0-9]|[$-_@.&+]|[!*\\(\\)
                ,]|(?:%[0-9a-fA-F][0-9a-fA-F]))+'
17          clean_text = re.sub(url_pattern, '', text)
18          return clean_text.strip()
19      return text
20
21  df = pd.read_csv("hf://datasets/giseldo/deep-se/deep-se.csv")
22
23  df = df[df['storypoint'] != 0]
24  df = df.dropna(subset=['storypoint', 'title', 'description'])
25
26  df['title'] = df['title'].apply(remove_html_tags)
27  df['description'] = df['description'].apply(remove_html_tags)
28
29  df['title'] = df['title'].apply(remove_urls)
30  df['description'] = df['description'].apply(remove_urls)
31
32  df['context'] = df['title'] + " " + df['description']
33
34  split_idx = int(len(df) * 0.7)
35  df = df.iloc[:split_idx]
36
37  train_df, test_df = train_test_split(df, test_size=0.2, random_state=42)
38
39  dataset = Dataset.from_pandas(pd.concat([train_df, test_df], keys=["train
        ", "test"], names=["split"]))
40  dataset = dataset.train_test_split(test_size=0.2)
41
42  model_name = "distilbert-base-uncased"
43  tokenizer = AutoTokenizer.from_pretrained(model_name)
```

```
44
45 def preprocess(example):
46     return tokenizer(example["context"], truncation=True, padding="
           max_length", max_length=128)
47
48 tokenized_datasets = dataset.map(preprocess)
49
50 tokenized_datasets = tokenized_datasets.map(lambda x: {"labels": float(x
       ["storypoint"])}, remove_columns=["storypoint", "context"])
51
52 model = AutoModelForSequenceClassification.from_pretrained(model_name,
       num_labels=1)
53
54 training_args = TrainingArguments(
55     output_dir="./results",
56     eval_strategy="epoch",
57     save_strategy="epoch",  # Add this line to match eval_strategy
58     learning_rate=2e-5,
59     per_device_train_batch_size=8,
60     per_device_eval_batch_size=8,
61     num_train_epochs=4,
62     weight_decay=0.01,
63     save_total_limit=1,
64     load_best_model_at_end=True,
65     metric_for_best_model="mse",
66 )
67
68 from sklearn.metrics import mean_squared_error
69
70 def compute_metrics(eval_pred):
71     predictions, labels = eval_pred
72     predictions = predictions.squeeze()
73     return {"mse": mean_squared_error(labels, predictions)}
74
75 trainer = Trainer(
76     model=model,
77     args=training_args,
```

```
78      train_dataset=tokenized_datasets["train"],
79      eval_dataset=tokenized_datasets["test"],
80      tokenizer=tokenizer,
81      compute_metrics=compute_metrics,
82  )
83
84  trainer.train()
85
86  trainer.save_model("./story_point_predictor")
```