Evaluating Machine Learning Algorithms for Software Effort Estimation from Textual Descriptions of Requirements

Rodrigo De Nadai Grigoleto[1], **Hilário Tomaz Alves de Oliveira**[1]

[1] Programa de Pós-Graduação em Computação Aplicada – PPComp
Instituto Federal do Espírito Santo (IFES)
Av. dos Sabiás 330 – Morada de Laranjeiras – Serra – ES – Brazil – 29166-630

rodrigogrigoleto@gmail.com, hilario.oliveira@ifes.edu.br

Abstract. *Effort estimation in software development is challenging, subjective, and time-consuming, and it directly impacts planning and resource allocation. This work investigated the effectiveness of Machine Learning (ML) algorithms and deep neural network architectures in automatically estimating development efforts from textual requirements descriptions. Traditional ML algorithms with TF-IDF representations, as well as BERT-based models, were evaluated on six public datasets. The experiments analyzed the performance of the models in two scenarios, intra- and inter-datasets, using the mean absolute error metric. The experimental results showed that BERT-based models outperformed traditional algorithms in the intra-dataset scenario, but experienced a more pronounced performance drop in the inter-dataset setting.*

Resumo. *A estimativa de esforço no desenvolvimento de software é uma tarefa desafiadora, subjetiva, que exige considerável empenho, impactando diretamente o planejamento e a alocação de recursos. Este trabalho investigou a eficácia de diferentes algoritmos de Aprendizado de Máquina (AM) e arquiteturas de redes neurais profundas na estimativa automática do esforço de desenvolvimento a partir de descrições textuais de requisitos. Foram avaliados algoritmos tradicionais de AM usando TF-IDF, bem como modelos baseados no BERT, em seis conjuntos de dados públicos. Os experimentos analisaram o desempenho dos modelos em dois cenários: intra e inter conjuntos de dados, utilizando a métrica de erro absoluto médio. Os resultados experimentais indicaram que os modelos baseados no BERT obtiveram o melhor desempenho no cenário intra bases de dados, mas apresentaram um declínio mais acentuado no cenário inter bases de dados em comparação com os algoritmos tradicionais.*

1. Introdução

A Engenharia de Software (ES) é uma área abrangente que engloba processos, métodos e ferramentas voltados ao desenvolvimento de sistemas e aplicações de *software* com qualidade [Pressman and Maxim 2021]. Um dos pilares fundamentais dessa área são os requisitos, que podem representar solicitações de novas funcionalidades, correção de erros ou modificações no *software* para agregar valor ao usuário. A interpretação desses documentos é uma tarefa desafiadora, pois exige uma análise semântica aprofundada e uma compreensão refinada da linguagem natural, especialmente para a estimativa do esforço necessário para sua implementação [Fernandes and Machado 2017].

O processo de desenvolvimento de *software*, marcado por rápidas mudanças tecnológicas e constantes alterações nas necessidades dos usuários, tem impulsionado a adoção de metodologias ágeis como um conjunto de práticas e princípios voltados à flexibilidade, colaboração e entrega incremental de valor [Massari 2018]. Essas metodologias permitem lidar de forma eficaz com demandas dinâmicas, que podem envolver desde a definição de novos requisitos até a correção de defeitos e a implementação de melhorias técnicas. Nesse contexto, a estimativa de esforço assume papel central, pois viabiliza o planejamento das atividades e a alocação adequada de recursos para atender às demandas priorizadas.

Entre as estratégias mais adotadas no âmbito ágil, destaca-se o uso dos *story points*, uma unidade de medida relativa que expressa o esforço necessário para implementar uma demanda específica. Diferentemente de métricas absolutas, os *story points* resultam de uma avaliação coletiva da equipe, considerando fatores como complexidade, volume de trabalho, riscos e incertezas. Essa definição é frequentemente realizada por meio de técnicas de estimativa relativa, como o *planning poker*, nas quais as tarefas são comparadas e pontuadas de acordo com a experiência do time, buscando consenso e minimizando subjetividades [Massari 2018].

A capacidade de estimar com precisão o esforço necessário para cada tarefa torna-se um fator determinante para o sucesso de um projeto. Estimativas imprecisas de esforço podem levar a estouros de orçamento, atrasos na entrega e falhas no projeto [Choetkiertikul et al. 2018]. Neste contexto, técnicas de Inteligência Artificial (IA) têm se mostrado promissoras para aprimorar a tomada de decisão em diversas atividades da ES, incluindo a estimativa de esforço [Alsaadi and Saeedi 2022]. Em particular, técnicas de Processamento de Linguagem Natural (PLN), combinadas com arquiteturas de redes neurais profundas, como demonstrado por Awad e Khanna [Awad and Khanna 2015], ou abordagens tradicionais de Aprendizado de Máquina (AM), conforme apresentado por Corrêa et al. [CORRÊA 2020], têm se destacado na automatização da interpretação textual, contribuindo para a análise e a estimativa de esforço no desenvolvimento de *software*.

Soluções baseadas na arquitetura *Transformers*, como o GPT2SP [Fu and Tantithamthavorn 2023], têm sido propostas para a estimativa de *story points* a partir de descrições textuais de requisitos. No entanto, modelos de linguagem de larga escala (LLMs, do inglês *Large Language Models*) demandam elevado poder computacional tanto para treinamento quanto para inferência, o que pode restringir sua aplicabilidade em contextos com recursos computacionais limitados. Nesse contexto, o uso de algoritmos tradicionais de aprendizado de máquina (AM) ou de modelos de linguagem pré-treinados (PLM, do inglês *Pre-trained Language Models*) pode representar uma alternativa viável, com menor custo computacional.

Corroborando essa perspectiva, estudos como o de Tawosi, Moussa e Sarro [Tawosi et al. 2024], que replicaram e analisaram criticamente o desempenho do GPT2SP, identificaram que, após correções em sua implementação, os resultados não superaram significativamente abordagens simples, como algoritmos de AM tradicionais. Isso evidencia a necessidade de investigações mais aprofundadas sobre a viabilidade do uso de modelos de AM para a estimativa de esforço em contextos ágeis. Além disso, poucos trabalhos exploram o desempenho desses modelos em cenários envolvendo diferentes bases de dados, ou seja, um modelo treinado em uma base de dados é capaz de manter seu

desempenho quando aplicado a um conjunto de dados distinto?

Este trabalho tem como objetivo investigar a eficácia de diferentes abordagens de aprendizado de máquina (AM) para a estimativa automática do esforço de desenvolvimento a partir de descrições textuais de requisitos de *software*. Para isso, foram analisados algoritmos tradicionais de AM, como o *Random Forest* aplicado a representações textuais baseadas na métrica *Term Frequency–Inverse Document Frequency* (TF-IDF), estratégias de combinação (*ensemble*) e modelos de linguagem pré-treinados (PLMs), como o *Bidirectional Encoder Representations for Transformers* (BERT) [Devlin et al. 2019]. Foram consideradas variantes do BERT com pré-treinamento geral (BERT base, BERT large, RoBERTa), além de modelos especializados no domínio de engenharia de *software*, como o SE-BERT e o BERT-SE. As estimativas foram realizadas em cenários que utilizam *story points*, sendo o problema modelado como uma tarefa de regressão.

As seguintes perguntas de pesquisa guiaram este trabalho:

> **Pergunta de Pesquisa 1 (PP1):** *Qual abordagem, AM tradicional com TF-IDF ou modelos neurais baseados em BERT, apresenta melhor desempenho na previsão do esforço de desenvolvimento de software a partir de descrições textuais?*
>
> **Pergunta de Pesquisa 2 (PP2):** *Qual é o desempenho de generalização dos algoritmos avaliados considerando um cenário de treinamento e teste em diferentes conjuntos de dados?*

Para responder às questões de pesquisa, foram realizados experimentos usando seis conjuntos de dados disponíveis publicamente [Fu and Tantithamthavorn 2023], e o desempenho dos algoritmos foi avaliado por meio da métrica de erro médio absoluto (MAE, do inglês *Mean Absolute Error*). Para avaliar a capacidade de generalização dos modelos, foram projetadas duas configurações experimentais: **(i)** um cenário intra conjunto de dados, no qual os modelos foram treinados, validados e testados no mesmo conjunto de dados utilizando validação cruzada com 5 subconjuntos (*5-fold cross-validation*); e **(ii)** um cenário inter conjunto de dados, no qual os modelos foram treinados em um conjunto de dados e testados em outro para avaliar sua capacidade de generalização. Com isto, este trabalho busca contribuir para o avanço do entendimento e da viabilidade de métodos de estimativa de esforço utilizando algoritmos de AM e PLN. O código-fonte desenvolvido está disponível publicamente em um repositório do GitHub[1].

2. Trabalhos Relacionados

Lidar com a estimativa do esforço para as diversas tarefas de um projeto de software emerge como uma tarefa que demanda grande trabalho por parte da equipe, momento pelo qual as variáveis, como o tempo e o preço do projeto, devem ser definidas. Segundo Pressman e Maxim [Pressman and Maxim 2021], fatores como a complexidade, tamanho e grau de incerteza do projeto podem inclusive afetar na confiabilidade das estimativas, acarretando situações indesejáveis como descumprimento de prazos de entrega, projeto inacabado, desperdícios de recursos, dentre outros problemas. Vários estudos com diversas abordagens diferentes vem contribuindo para um

[1] `https://github.com/projetomestradoifes/ml-esforco-projetos-software`

constante aperfeiçoamento de técnicas de estimativas de esforço de tarefas de desenvolvimento de software, utilizando algoritmos de AM e PLN, apresentando diversas possibilidades para lidar com a complexidade e a ambiguidade dos requisitos textuais [Choetkiertikul et al. 2018, Alsaadi and Saeedi 2022, Tawosi et al. 2024].

Jadhav et al. [Jadhav et al. 2024] introduziram uma abordagem baseada em seleção dinâmica de *ensembles* (MS-DES) para estimar esforço em projetos de software. Essa técnica combina múltiplos modelos preditivos e seleciona dinamicamente os mais adequados para cada conjunto de dados. O estudo demonstrou que a abordagem não apenas melhorou a precisão das estimativas, mas também reduziu a complexidade computacional, tornando-se uma solução eficiente para cenários reais. Além disso, o uso de técnicas de seleção de atributos contribuiu para identificar as características mais relevantes nos conjuntos de dados, garantindo resultados consistentes em diferentes contextos.

No trabalho desenvolvido por Atoum e Otoom [Atoum and Otoom 2024] foi proposta uma abordagem baseada no uso de *word embeddings* pré-treinados, como *FastText* e o GPT-2, para melhorar a estimativa de esforço em conjuntos de dados pequenos. Utilizando modelos tradicionais de AM, como *Support Vector Machines* e *Random Forests*, os autores demonstraram que é possível alcançar desempenho comparável ao de modelos de aprendizado profundo, como o GPT-2, com menor custo computacional e maior interpretabilidade. O estudo destacou a eficácia do uso de *embeddings* contextualizados para capturar nuances semânticas nas descrições de requisitos, mostrando-se uma alternativa viável e eficiente ao uso de técnicas mais complexas em contextos com dados limitados.

Fu e Tantithamthavorn [Fu and Tantithamthavorn 2023] propuseram o modelo *GPT2SP*, baseado na arquitetura *Generative Pre-trained Transformers* (GPT), para a estimativa de esforço em projetos ágeis, destacando-se por sua capacidade de capturar o contexto semântico de descrições textuais complexas. No entanto, uma replicação do estudo conduzida por Tawosi, Moussa e Sarro [Tawosi et al. 2024] identificou falhas na avaliação original do GPT2SP, incluindo erros na computação das métricas de desempenho, que inflacionaram seus resultados. Após corrigirem essas inconsistências, os pesquisadores demonstraram que o GPT2SP não apresenta um desempenho significativamente superior a abordagens mais simples, como o uso da mediana do esforço de histórias anteriores. Além disso, o estudo de Tawosi et al. também realizou uma replicação rigorosa do modelo *Deep-SE*, proposto originalmente por Choetkiertikul et al. [Choetkiertikul et al. 2018]. Embora o *Deep-SE* tenha apresentado bons resultados no estudo original, a replicação revelou que ele não superou significativamente algoritmos de AM tradicionais usados como *baselines* em muitos casos, inclusive sendo inferior em cenários de estimativa entre projetos. Esses achados reforçam a necessidade de avaliações rigorosas na aplicação de modelos de aprendizado profundo para estimativa de esforço, evidenciando que desafios persistem na busca por soluções mais eficazes e generalizáveis para projetos ágeis.

A Tabela 1 resume os principais trabalhos relacionados, detalhando as bases de dados e os algoritmos empregados. Observa-se que os estudos revisados abrangem desde algoritmos tradicionais de AM até o uso do GPT-2, considerando diferentes bases de dados. Contudo, esses trabalhos apontam duas direções de pesquisa futura como fundamentais para o avanço da área: **(i)** uma investigação mais aprofundada sobre a capacidade de generalização das diferentes abordagens de AM; e **(ii)** a exploração de modelos neurais desenvolvidos especificamente para o domínio da engenharia de software.

Tabela 1. Resumo comparativo dos trabalhos relacionados.

Trabalho	Base de dados	Algoritmos
[Jadhav et al. 2024]	Bases de dados públicas de empresas de TI.	Algoritmos tradicionais de regressão, como *Naive Bayes* e *Random Forest*; algoritmos baseados em *ensemble*, como o *MS-DES*.
[Atoum and Otoom 2024]	Derivada do conjunto Deep-SE, com 16 projetos open-source.	*Support Vector Machines*, *Random Forest*, *XGBoost* e outros algoritmos tradicionais de regressão combinados com embeddings pré-treinados como FastText, SBERT e GPT-2.
[Tawosi et al. 2024]	Derivada de projetos públicos disponíveis no Jira.	GPT2 e Deep-SE.
[Fu and Tantithamthavorn 2023]	Derivada de projetos públicos disponíveis no Jira.	GPT2.
Trabalho proposto	Derivada de projetos públicos disponíveis no Jira.	Algoritmos tradicionais de regressão, como o *Random Forest*; algoritmos de ensemble, como DesReg; modelos baseados no BERT.

Este trabalho expande pesquisas anteriores ao analisar diferentes abordagens de AM, abrangendo desde métodos baseados em algoritmos tradicionais de regressão e estratégias de *ensemble* combinadas com a métrica TF-IDF até modelos baseados no BERT, incluindo versões pré-treinadas de uso geral e especializadas no domínio da engenharia de *software*. Além disso, investigamos o desempenho dos modelos em dois cenários distintos: **(i)** *intra base de dados*, em que o treinamento e a avaliação são realizados na mesma base, permitindo a análise do ajuste do modelo aos dados originais; e **(ii)** *inter base de dados*, em que o modelo é treinado em uma base e avaliado em outra, possibilitando a verificação de sua capacidade de generalização para novos contextos.

3. Materiais e Métodos

Nesta seção, são apresentadas as bases de dados utilizadas nos experimentos (Subseção 3.1), as abordagens de Aprendizado de Máquina (AM) investigadas (Subseção 3.2) e, por fim, a metodologia de avaliação usada nos experimentos (Subseção 3.3).

3.1. Bases de Dados

Neste trabalho, foram utilizadas seis bases de dados, todas disponibilizadas publicamente no GitHub[2] e previamente adotadas em outros estudos [Fu and Tantithamthavorn 2023, Tawosi et al. 2024]. As estimativas dessas bases de dados são baseadas em *story points* e pertencem aos projetos *Talend Data Quality*, *Spring XD*, *Moodle*, *Mesos*, *Data Management* e *Appcelerator Studio*. Cada registro das bases de dados contém as seguintes informações: **(i)** *id*, identificador único de cada registro; **(ii)** *título*, correspondente ao nome da tarefa no repositório; **(iii)** *descrição*, que detalha a tarefa e o trabalho a ser realizado; e **(iv)** *story point*, que representa o valor do esforço estimado para a tarefa.

Na Tabela 2 são apresentas as seguintes estatísticas descritivas das bases de dados: quantidade de tarefas (*issues*), média e desvio padrão do número de palavras por registro, além dos valores mínimo, máximo, médio e mediano da unidade de esforço (*story point*).

[2] `https://github.com/awsm-research/gpt2sp/tree/main/sp_dataset/marked_data`

Tabela 2. Estatísticas das bases de dados utilizadas.

Base de Dados	Total Registros	Média Palavras	Desv. Padrão	SP_{Min}	SP_{Max}	$SP_{Média}$	$SP_{Mediana}$
Appcelerator Studio	2.689	98,56	119,38	1	8	5,08	5
Spring XD	2.903	70,75	104,19	1	8	3,43	3
Data Management	2.992	66,33	88,71	1	8	3,17	2
Moodle	678	70,53	68,24	1	8	4,49	5
Mesos	1.526	163,46	384,72	1	8	2,84	3
Talend Data Quality	937	78,58	90,98	1	8	4,39	5

Na etapa de pré-processamento, os registros de cada base de dados foram filtrados para incluir apenas aqueles com descrições escritas em inglês. Além disso, registros duplicados, descrições nulas ou compostas apenas por espaços em branco foram removidos para evitar redundâncias e entradas inválidas. Nos conjuntos de dados utilizados neste trabalho, os *story points* são representados por valores numéricos inteiros e crescentes, nos quais números maiores indicam maior esforço estimado para a realização da tarefa. Originalmente, as bases continham valores variando de 0 até 100; no entanto, observou-se que valores superiores a 8 eram extremamente raros e pouco representativos para a maioria dos projetos analisados. Por esse motivo, foi adotada uma estratégia de filtragem baseada em um limite fixo, removendo registros com *story points* superiores a 8. Assim, os dados analisados contemplam majoritariamente os valores 1, 2, 3, 5 e 8, alinhando-se à prática comum de estimativas crescentes, nas quais o valor maior está associado a maior esforço. Essa abordagem foi escolhida para minimizar o impacto de valores atípicos que poderiam distorcer a modelagem preditiva, garantindo maior consistência na distribuição dos dados sem comprometer a representatividade das estimativas.

Na Tabela 3 são apresentados exemplos de descrições de tarefas extraídas respectivamente das bases de dados do *Data Management*, *Moodle* e *Mesos*. Cada registro contém o identificador da tarefa, seu título, a descrição original e o respectivo valor de *story point* (SP).

Tabela 3. Exemplos de descrições de tarefas extraídas das bases de dados.

ID	Título	Descrição Original	SP
DM-3947	Remove dependency on mysqldb in wmgr	Move remaining code that depends on mysqldb to use sqlalchemy instead. Ensure all database interactions are compatible with the updated framework.	1
MESOS-4222	Document containerizer from user perspective	Add documentation that covers: the purpose of the containerizer, how to configure it, usage examples, and best practices from the user's perspective. This helps new users understand and apply the tool effectively.	3
MDL-44128	Atto: Toolbar menus should work with keyboard navigation	It is confusing when using left-right through the Atto toolbar with a keyboard. Menu selections and navigation should follow accessibility standards to ensure consistent and intuitive keyboard use.	8

3.2. Abordagens de AM Avaliadas

Neste trabalho, avaliamos duas abordagens de aprendizado de máquina para a estimativa de *story points* a partir da descrição textual dos requisitos de software. A primeira abordagem emprega algoritmos tradicionais de regressão, enquanto a segunda utiliza modelos baseados em redes neurais profundas com a arquitetura *Transformer* [Vaswani et al. 2017], especificamente modelos derivados do *Bidirectional Encoder Representations for Transformers* (BERT) [Devlin et al. 2019].

Os modelos tradicionais de AM utilizados incluem algoritmos simples e métodos de *ensemble*. Os algoritmos considerados foram *Ridge Regression*, *Lasso Regression*, *ElasticNet Regression*, *Support Vector Regression* (SVR), *K-Nearest Neighbors* (KNN), *Decision Tree*, *Random Forest* (da biblioteca Scikit-learn[3]), além dos algoritmos XGBoost[4] e LightGBM[5]. Também foram avaliados dois métodos de *ensemble*: o tradicional *Stacking* e o algoritmo *Dynamic Ensemble Selection for Regression* (DESRegression)[6] [Pérez-Godoy et al. 2024]. Esses algoritmos foram treinados com representações das descrições dos requisitos, obtidas por meio da medida de *Term Frequency-Inverse Document Frequency* (TF-IDF), que atribui pesos às palavras com base na sua frequência relativa no conjunto de dados.

Este trabalho também utilizou modelos baseados no BERT, uma arquitetura de rede neural profunda que emprega mecanismos de atenção para capturar o contexto bidirecional das palavras em uma sentença, permitindo uma compreensão mais aprofundada das nuances linguísticas [Devlin et al. 2019]. Os modelos considerados foram BERT base e BERT large, RoBERTa base e RoBERTa large, CodeBERT, SE-BERT e BERT para engenharia de software (BERT-SE), todos disponíveis publicamente na plataforma Hugging Face[7]. Os modelos SE-BERT e BERT-SE são especializados no domínio da engenharia de *software*, com o objetivo de capturar terminologias e contextos específicos dessa área. O modelo CodeBERT é uma versão do BERT treinada com dados envolvendo linguagem natural e linguagem de programação. O objetivo é avaliar se o uso de modelos com pré-treinamento específico para engenharia de *software* resulta em melhor desempenho em comparação com modelos com pré-treinamento geral.

As descrições textuais foram submetidas a um processo de limpeza, no qual caracteres inválidos e elementos na linguagem *Hypertext Markup Language* (HTML) foram removidos. Para os modelos tradicionais de AM, foram aplicadas etapas adicionais, incluindo lematização, remoção de *stopwords* e símbolos de pontuações. Essa abordagem reduziu o ruído nas descrições das tarefas, preservando os elementos mais relevantes para a estimativa de esforço. Por outro lado, para os modelos baseados em BERT, foi mantida apenas a etapa inicial de limpeza.

3.3. Metodologia Experimental

A metodologia experimental utilizada neste trabalho foi planejada para responder às perguntas de pesquisa que orientaram os experimentos realizados. É importante destacar que

[3]https://scikit-learn.org
[4]https://xgboost.readthedocs.io
[5]https://lightgbm.readthedocs.io/
[6]https://github.com/lperezgodoy/DESReg
[7]https://huggingface.co/

as bases de dados utilizadas neste estudo não são fornecidas com divisões padronizadas entre conjuntos de treinamento, validação e teste. Dessa forma, cada autor define sua própria estratégia de particionamento, o que torna inviável a comparação direta dos resultados obtidos com aqueles reportados em estudos anteriores. Para permitir uma avaliação justa e internamente consistente, adotamos uma metodologia experimental própria, aplicada de forma uniforme a todos os modelos analisados.

A **PP1** investiga qual abordagem, entre AM tradicional com representações baseadas em TF-IDF e modelos neurais baseados no BERT, apresenta melhor desempenho na estimativa do esforço de desenvolvimento de software a partir de descrições textuais. Para responder a essa questão, foi realizado um primeiro experimento (Cenário intra base de dados), no qual diferentes modelos foram treinados e avaliados, utilizando a métrica de erro absoluto médio (MAE). O MAE foi adotado por ser a métrica mais comumente usada nos trabalhos relacionados [Fu and Tantithamthavorn 2023, Tawosi et al. 2024]. Para cada uma das seis bases de dados, foi utilizada a metodologia de validação cruzada (*k-fold*) com 5 divisões. Em cada iteração, 10% do conjunto de treinamento foi separado para ser usado como conjunto de validação. Essa análise permitiu identificar qual abordagem foi mais eficaz na extração de padrões textuais relevantes para a previsão de *story points* em cada base de dados.

A segunda pergunta de pesquisa, **PP2**, busca avaliar a capacidade de generalização dos modelos quando treinados e testados em diferentes conjuntos de dados. Para isso, foi realizado um experimento no qual, a cada rodada, um conjunto de dados foi utilizado para teste, outro para validação e os quatro restantes para treinamento, repetindo o procedimento até que cada base fosse utilizada em todas as funções. Assim, é possível verificar se os modelos mantêm desempenho diante de diferentes estilos de escrita, granularidade das descrições e critérios subjetivos presentes em projetos distintos. Cabe ressaltar que, embora os pontos de história possam ser definidos de forma diferente por cada equipe, neste experimento a análise considera exclusivamente as informações textuais dos requisitos, desconsiderando as experiências e convenções individuais dos times. O objetivo é investigar a robustez dos modelos a partir dos padrões linguísticos presentes nas descrições, mesmo diante das particularidades de cada base de dados.

Em todos os experimentos, os algoritmos de AM tradicionais foram treinados usando palavras (unigramas) para compor o vocabulário, com um limite máximo de 5.000 palavras. Esse valor foi escolhido com base no custo-benefício entre desempenho e custo computacional. Já os modelos baseados em BERT passaram por etapas de ajuste fino (*fine-tuning*) por no máximo 15 épocas. Para evitar sobreajuste, foram adotadas as técnicas de parada antecipada (*early stopping*) e monitoramento do treinamento. Ao final de cada época, o modelo resultante foi aplicado no conjunto de validação e a métrica de MAE foi computada. Apenas o modelo com o menor MAE ao longo do treinamento foi salvo e aplicado no conjunto de teste. Caso o MAE não apresentasse redução ao longo de três épocas, o treinamento foi interrompido.

4. Resultados

Nesta seção são apresentados os resultados dos experimentos realizados para responder às questões de pesquisa **PP1** e **PP2**. Inicialmente, na Subseção 4.1 são apresentados os resultados do cenário intra base de dados, enquanto na Subseção 4.2 são discutidos os

resultados do cenário inter base de dados.

4.1. Experimento 1 - Cenário intra conjunto de dados

Na Tabela 4, são apresentados os resultados deste primeiro experimento, considerando o cenário intra bases de dados, com base na medida do MAE. O modelo com o menor valor de MAE em cada conjunto de dados e o melhor modelo geral, considerando a média dos valores de MAE, estão destacados em negrito.

Tabela 4. Resultados do experimento no cenário intra conjunto de dados.

Abordagem	Modelo	Appcelerator Studio	Spring XD	Data Management	Moodle	Mesos	Talend Data Quality	Média
Tradicionais	Decision Tree	1,746	1,702	1,809	2,033	1,404	2,363	1,843
	ElasticNet	1,619	1,790	1,797	2,113	1,313	2,260	1,815
	KNN	1,735	1,681	1,659	2,003	1,362	2,258	1,783
	Lasso	1,619	1,790	1,797	2,115	1,313	2,260	1,816
	LightGBM	1,670	1,592	1,649	1,970	1,285	2,203	1,728
	Random Forest	1,594	1,549	1,570	1,891	1,214	2,121	1,657
	Ridge	1,658	1,582	1,585	1,937	1,237	**2,118**	1,686
	SVR	1,595	1,571	1,568	1,947	1,198	2,144	1,671
	XGBoost	1,689	1,599	1,645	1,932	1,311	2,262	1,740
	DESRegression	1,614	1,579	1,593	1,943	1,262	2,141	1,689
	Stacking	1,598	1,550	1,578	1,896	1,221	2,122	1,661
BERT	BERT Base	1,596	1,511	1,481	**1,880**	1,222	2,185	1,646
	BERT Large	1,630	1,630	1,608	2,060	1,198	2,249	1,729
	RoBERTa Base	1,609	1,516	1,517	1,936	1,213	2,235	1,671
	RoBERTa Large	1,618	1,725	1,781	2,101	1,301	2,257	1,797
	CodeBERT	1,583	1,542	1,534	1,964	1,235	2,196	1,675
	SE-BERT	**1,511**	**1,444**	1,449	1,952	**1,148**	2,138	**1,607**
	BERT-SE	1,558	1,479	**1,434**	1,939	1,152	2,229	1,632

Os modelos baseados no BERT apresentaram os melhores desempenhos em comparação com os modelos tradicionais. O SE-BERT obteve os menores valores de erro na medida do MAE em três bases de dados (*Appcelerator Studio*, *Spring XD* e *Meso*), enquanto os modelos BERT-SE e BERT Base alcançaram os menores valores de MAE nas bases *Data Management* e *Moodle*, respectivamente. A única exceção foi observada na base *Talend Data Quality*, na qual o modelo Ridge apresentou o melhor desempenho. Considerando a média dos valores de MAE nas seis bases de dados, os três modelos com melhor desempenho foram SE-BERT, BERT-SE e BERT Base. Considerando apenas os modelos tradicionais, o *Random Forest* apresentou o melhor desempenho médio, sendo competitivo principalmente nos conjuntos *Moodle* e *Talend Data Quality*. Além desse algoritmo, destaca-se o desempenho do método de combinação Stacking, que obteve a segunda menor taxa de erro entre os modelos tradicionais.

Com base nos resultados deste experimento, alguns pontos podem ser destacados. Primeiro, os modelos especializados BERT-SE e SE-BERT, voltados para o domínio da engenharia de software, apresentaram os melhores resultados, evidenciando o impacto positivo da especialização de domínio, especialmente em cenários com vocabulário altamente específico. Diferentemente dos demais modelos avaliados, o CodeBERT foi o único treinado com dados de linguagens de programação. No entanto, essa exposição a código não teve um impacto significativo nos resultados, uma vez que seu desempenho foi inferior ao dos modelos do BERT base e RoBERTa base. Além disso, o uso das arquiteturas maiores (*large*) dos modelos BERT e RoBERTa não resultou em um desempenho

superior em comparação com a arquitetura *base*. Esse resultado pode estar relacionado à quantidade limitada de registros nas bases de dados utilizadas, já que todas possuem menos de 3.000 exemplos, enquanto arquiteturas de redes neurais maiores exigem mais dados para um treinamento eficaz. Por fim, é importante destacar que, embora os modelos baseados no BERT tenham apresentado melhores resultados em comparação com os algoritmos tradicionais de AM, seu uso implica um custo computacional mais elevado tanto para treinamento quanto para inferência.

4.2. Experimento 2 - Cenário inter conjunto de dados

Neste experimento, a capacidade de generalização dos modelos foi avaliada, sendo o treinamento realizado em uma base de dados e os testes conduzidos em outra distinta. Essa metodologia foi adotada para analisar o desempenho dos modelos diante de variações no vocabulário, no estilo de escrita e na distribuição de *story points* entre diferentes projetos. Os resultados deste segundo experimento são apresentados na Tabela 5, utilizando a métrica de erro médio absoluto (MAE). Os modelos com menor valor de MAE em cada base de dados, assim como o melhor desempenho geral, estão destacados em negrito.

Tabela 5. Resultados do experimento no cenário inter conjunto de dados.

Abordagem	Modelo	Appcelerator Studio	Spring XD	Data Management	Moodle	Mesos	Talend Data Quality	Média
Tradicionais	Decision Tree	2,309	1,960	1,872	2,223	1,922	2,304	2,098
	ElasticNet	2,249	1,997	1,870	2,157	2,043	2,291	2,101
	KNN	2,429	2,246	1,903	2,143	1,940	2,349	2,168
	Lasso	2,249	1,997	1,870	2,157	2,043	2,291	2,101
	LightGBM	2,374	1,875	1,777	2,207	1,829	2,332	2,065
	Random Forest	2,444	1,850	1,734	2,252	1,671	2,395	2,058
	Ridge	2,431	1,991	1,856	2,192	1,829	2,289	2,098
	SVR	2,542	1,831	**1,688**	2,200	1,699	2,317	2,046
	XGBoost	2,370	1,874	1,803	2,208	1,882	2,321	2,076
	DESRegression	2,352	2,051	1,821	**2,122**	1,863	**2,276**	2,081
	Stacking	2,255	1,912	1,829	2,167	1,864	2,305	2,055
BERT	BERT Base	**2,132**	**1,719**	1,861	2,258	1,370	2,377	**1,953**
	BERT Large	2,211	1,979	1,883	2,290	1,831	2,278	2,079
	RoBERTa Base	2,133	1,766	1,892	2,238	1,491	2,287	1,968
	RoBERTa Large	2,169	1,884	1,894	2,178	1,874	2,285	2,047
	CodeBERT	2,356	1,813	2,080	2,383	1,729	2,300	2,110
	SE-BERT	2,367	1,742	2,176	2,354	**1,331**	2,335	2,051
	BERT-SE	2,183	1,819	2,032	2,381	1,697	2,350	2,077

Diferentemente do cenário intra bases de dados, os modelos baseados no BERT não apresentaram, de forma consistente, os melhores resultados. Embora o melhor desempenho médio geral tenha sido registrado pelo BERT Base, os modelos tradicionais demonstraram desempenho competitivo. Ainda assim, um desempenho superior foi observado pelos modelos baseados em BERT em alguns contextos. Os menores valores de MAE foram obtidos nos conjuntos de dados *Spring XD* e *Appcelerator Studio* com o uso do BERT Base, enquanto o SE-BERT apresentou o melhor desempenho na base de dados *Mesos*, reforçando o potencial dos modelos específicos de domínio. O menor valor de MAE no conjunto de dados *Data Management* foi alcançado pelo algoritmo SVR, enquanto o algoritmo DESRegression destacou-se ao obter o melhor desempenho nas bases de dados *Moodle* e *Talend Data Quality*. De modo geral, foi observado um aumento na taxa de erro em comparação com os resultados do Experimento 1. Esse aumento nos

valores de MAE variou de 13,851% (KNN) a 27,267% (BERT-SE). Essa redução no desempenho dos modelos reflete o desafio imposto pelo cenário entre diferentes bases de dados, que se assemelha mais ao ambiente real.

Os resultados experimentais evidenciam que, embora os modelos BERT Base e RoBERTa Base tenham alcançado os menores valores médios de MAE, a melhoria em relação aos modelos tradicionais não foi tão acentuada quanto no cenário intra conjunto de dados. Um possível motivo é o tamanho reduzido dos conjuntos de dados utilizados, o que pode ter limitado o desempenho dos modelos maiores, como BERT-large e RoBERTa-large, que geralmente exigem volumes maiores de dados de treinamento para alcançar seu potencial. Além disso, a variação nos melhores desempenhos entre os diferentes conjuntos de dados reforça a importância da escolha do modelo conforme as características do domínio específico. Assim, futuros trabalhos podem explorar técnicas de adaptação de domínio ou novos métodos de combinação (*ensemble*) para mitigar a perda de desempenho observada no cenário de generalização entre bases de dados.

5. Considerações Finais e Trabalhos Futuros

Este trabalho avaliou o desempenho de diferentes algoritmos tradicionais de aprendizado de máquina e variantes baseadas no modelo *BERT* para a estimativa do esforço de desenvolvimento a partir de descrições textuais. Os experimentos foram conduzidos em dois cenários: **(i)** *intra conjunto de dados*, no qual o modelo *SE-BERT* apresentou a menor taxa de erro na métrica MAE; e **(ii)** *inter conjunto de dados*, em que o modelo *BERT Base* obteve a menor média de erro, embora os modelos tradicionais tenham demonstrado desempenho competitivo. Os resultados experimentais indicam que modelos baseados em *BERT*, especialmente aqueles especializados no domínio de engenharia de *software*, são mais eficazes quando treinados e avaliados no contexto de um mesmo projeto. No entanto, em cenários de generalização entre projetos distintos, modelos tradicionais, como *SVR* e o método de combinação (*ensemble*) de algoritmos do DESRegression, apresentaram desempenho próximo ou superior em algumas situações, configurando-se como alternativas viáveis em ambientes com maior heterogeneidade textual.

Como trabalhos futuros, propõe-se a investigação de estratégias para aprimorar a capacidade de generalização dos modelos baseados no BERT, como o ajuste fino em múltiplas bases de dados e o uso de técnicas de adaptação entre domínios. Além disso, a utilização de ferramentas como o Optuna[8] pode contribuir para a otimização dos hiperparâmetros dos algoritmos. Por fim, a incorporação de metadados relacionados à tarefa de estimativa de esforço, como a experiência da equipe e as características do projeto, configura-se como uma abordagem promissora para enriquecer a representação textual e aprimorar a precisão das estimativas em cenários reais, ao integrar informações contextuais sobre a equipe e o projeto.

Agradecimentos

Os autores agradecem a FAPES/UnAC (Nº FAPES 1228/2022 P 2022-CD0RQ, Nº SIAFEM 2022-CD0RQ) pelo apoio financeiro dado por meio do Sistema UniversidaES.

[8] `https://optuna.org/`

Referências

Alsaadi, B. and Saeedi, K. (2022). Data-driven effort estimation techniques of agile user stories: a systematic literature review. *Artificial Intelligence Review*, 55(7):5485–5516.

Atoum, I. and Otoom, A. A. (2024). Enhancing software effort estimation with pre-trained word embeddings: A small-dataset solution for accurate story point prediction. *Electronics*, 13(23).

Awad, M. and Khanna, R. (2015). *Efficient Learning Machines: Theories, Concepts, and Applications for Engineers and System Designers*. Apress.

Choetkiertikul, M., Dam, H. K., Tran, T., Pham, T., Ghose, A., and Menzies, T. (2018). A deep learning model for estimating story points. *IEEE Transactions on Software Engineering*, 45(7):637–656.

CORRÊA, W. A. (2020). Aplicando aprendizado de máquina para estimativa de esforço no desenvolvimento de software. Master's thesis. DEPARTAMENTO DE INFORMÁTICA/CCET.

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. (2019). Bert: Pre-training of deep bidirectional transformers for language understanding. In *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.

Fernandes, J. M. and Machado, R. J. (2017). *Requisitos em projetos de software e de sistemas de informação (Portuguese Edition)*. Novatec Editora. Edição do Kindle., 1ª edição edition.

Fu, M. and Tantithamthavorn, C. (2023). Gpt2sp: A transformer-based agile story point estimation approach. *IEEE Transactions on Software Engineering*, 49(2):611–625.

Jadhav, A., Shandilya, S. K., Izonin, I., and Muzyka, R. (2024). Multi-step dynamic ensemble selection to estimate software effort. *Applied Artificial Intelligence*, 38(1):2351718.

Massari, V. (2018). *Gerenciamento Ágil de Projetos (2a. edição)*. Brasport.

Pressman, R. S. and Maxim, B. R. (2021). *Engenharia de software*. McGraw Hill Brasil, 9ª edição edition.

Pérez-Godoy, M. D., Molina, M., Martínez, F., Elizondo, D., Charte, F., and Rivera, A. J. (2024). Desreg: Dynamic ensemble selection library for regression tasks. *Neurocomputing*, 580:127487.

Tawosi, V., Moussa, R., and Sarro, F. (2024). Agile effort estimation: Have we solved the problem yet? insights from the replication of the gpt2sp study. In *2024 IEEE International Conference on Software Analysis, Evolution and Reengineering (SANER)*, pages 1034–1041.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. (2017). Attention is all you need. *Advances in neural information processing systems*, 30.