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

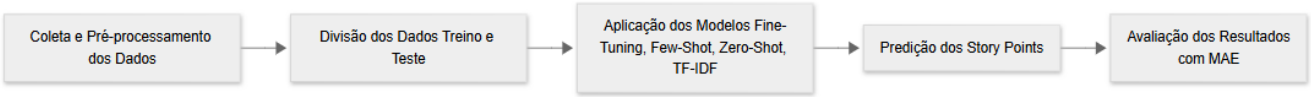

Figura 1: Procedimentos realizados

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

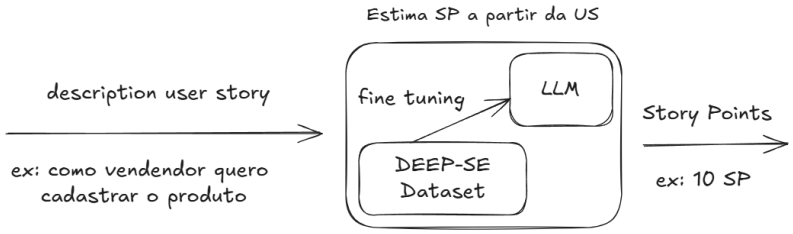

Figura 2: Arquitetura alto nível da proposta

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.

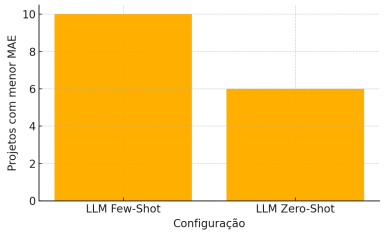

Figura 3: Desempenho: Few-shot vs Zero Shot

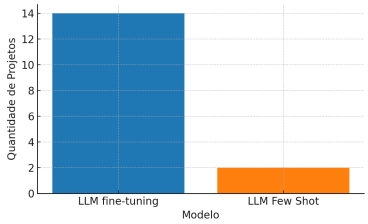

Figura 4: Desempenho: Fine-tuning vs Few-Shot

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.

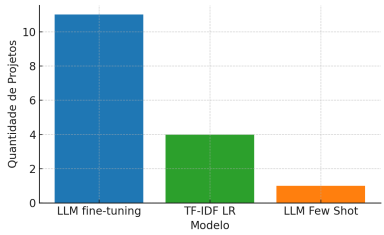

Figura 5: Desempenho: Todos

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