Journal of the Brazilian Computer Society, 2026, 32:1, doi: 10.5753/jbcs.2026.5860

User Story Estimation using Natural Language Processing and Deep Learning: A Comparative Study

Daniel de Oliveira Silva [**Federal University of Technology – Paraná - Cornélio Procópio** | ***daniels.1998@alunos.utfpr.edu.br***]
Alinne Cristinne Corrêa Souza [**Federal University of Technology – Paraná - Dois Vizinhos** | ***alinnesouza@utfpr.edu.br***]
Francisco Carlos Monteiro Souza [**Federal University of Technology – Paraná - Dois Vizinhos** | ***franciscosouza@utfpr.edu.br***]
Silvio Ricardo Rodrigues Sanches [**Federal University of Technology – Paraná - Cornélio Procópio** | ***silviosanches@utfpr.edu.br***]

Graduate Program in Informatics (PPGI), Federal University of Technology – Paraná (UTFPR), Cornélio Procópio Campus, Av. Alberto Carazzai, 1640, Centro, Cornélio Procópio, Paraná, 86300-000, Brazil.

Received: 08 April 2025 • **Accepted:** 26 January 2026 • **Published:** 24 April 2026

Abstract Effort estimation is a fundamental activity in software development, guiding task prioritization, resource allocation, and cost planning. Traditional techniques, such as Planning Poker, rely heavily on team subjectivity and experience, which can compromise their effectiveness in certain contexts. This study explores how machine learning and natural language processing techniques can increase the accuracy of effort estimation for user stories. To understand the current state of research in this area, a systematic mapping was conducted to identify the main databases, techniques, and methods used to estimate effort based on textual requirements. Guided by the mapping, a comparative experimental study was performed using the FastText and XLNet language models, combined with a deep neural network, on a dataset of 23,313 user stories. The results indicate that XLNet outperformed FastText in most evaluation metrics, achieving a Mean Absolute Error (MAE) of 3.77, a Mean Squared Error (MSE) of 79.94, and a Median Absolute Error (MdAE) of 1.93. Furthermore, the proposed approach performed competitively compared to related works. These findings demonstrate the potential of deep learning models to assist developers by providing more accurate, consistent, and objective estimates for user story effort.

Keywords: user story, effort estimation, natural language processing, deep learning

1 Introduction

Requirements engineering is a fundamental phase of software engineering and is responsible for specifying, documenting, and maintaining the system's requirements. This phase involves activities such as elicitation, analysis, and modeling. Software requirements are detailed and specific descriptions of the functionalities, behaviors, and constraints of a system, that is, they define what the software must do and how it should behave under various conditions [SOMMERVILLE, 2004]. This stage plays a critical role in defining the project scope, planning, and resource allocation. In agile practices, collaborative techniques such as Planning Poker are widely used to support effort estimation.

Effort estimation is a key part of requirements engineering, aiming to predict the amount of effort required to implement a given requirement. One of the main challenges in software development is inaccurate effort estimation, which can lead to poor quality, delivery delays, underqualified staff, and budget overruns [Bhat *et al*., 2006].

According to the report [Jobera Editorial Team, 2025], approximately 52.7% of software projects exceed their original budgets by at least 89%, and 47% of unsuccessful projects fail primarily due to poor project management. Furthermore, ineffective requirements management accounts for 32% of software project failures.

Effort estimation typically involves collaboration between project managers and development teams to assess the complexity of software requirements, such as technical tasks or user stories. Various techniques are used to support this estimation, with the most common being expert judgment (e.g., Planning Poker) and algorithmic or parametric approaches (e.g., COCOMO, Function Points).

Planning Poker is a collaborative technique in which the development team meets to discuss and assign a score to each user story. These scores, known as story points, represent the perceived difficulty of implementing each requirement. The technique encourages active participation, promoting discussion about the reasoning behind each estimate, particularly for extreme values. This collaborative dynamic helps align the team's understanding of task complexity [Cohn, 2005].

Despite its popularity, Planning Poker has limitations that may affect its effectiveness. It relies heavily on the individual knowledge and experience of team members. Accurate estimation requires familiarity with the requirement, which may not always be the case. This human dependency can result in inaccurate estimates, particularly in complex projects or when team experience levels vary [Aranda and Easterbrook, 2005].

In recent years, advances in machine learning (ML) have

opened new possibilities for automating effort estimation in user stories. Recent studies have demonstrated the potential of ML, based models to predict effort based on textual requirements, showing promising results [Choetkiertikul *et al*., 2018; Gultekin and Kalipsiz, 2020; Ramessur and Nagowah, 2021]. Moreover, the use of pre-trained Natural Language Processing (NLP) models, such as BERT (Bidirectional Encoder Representations for Transformers) [Fávero *et al*., 2022], GPT-3 (Generative Pre-trained Transformer 3) [Baratto, 2022], and HAN (Hierarchical Attention Network) [Kassem *et al*., 2023], has proven to be particularly effective, enhancing the analysis and interpretation of complex textual data.

To investigate machine learning techniques and models for effort estimation from textual requirements, this study follows two complementary research paths.

Conducting an extensive literature search, no previous systematic mapping has specifically addressed the use of machine learning techniques for software effort estimation. Although [Fávero *et al*., 2018] conducted a systematic mapping focused on analogy-based software effort estimation techniques, the scope of their study was restricted to approaches that derive estimates from historical cases (e.g., ANFIS, collaborative filtering, and radial basis functions). Covering the period from 2007 to 2017, their work highlighted trends and gaps exclusively in analogy-based methods, which opened the opportunity for a broader mapping focused on machine learning techniques applied to textual requirements.

Regarding experimentation, previous studies, such as [Choetkiertikul *et al*., 2018], [Fávero *et al*., 2022], and [Kassem *et al*., 2023], have already applied language models combined with deep learning architectures to estimate software effort. Building on this line of research, the present study aims to evaluate additional language models, specifically FastText (uncontextualized) and XLNet (contextualized), on large-scale user story datasets. The goal is to investigate whether alternative embeddings can improve predictive performance, also considering computational cost and fine-tuning effects.

To guide this research, two levels of research questions were defined:

- **GRQ1:** Identify which approaches have applied machine learning to software effort estimation.
- **GRQ2:** Assess the potential of using language models to enhance these approaches.

Specific research questions were defined for each substudy. In the **systematic mapping**, the questions focus on the characteristics of the datasets used, the algorithms, techniques, and evaluation metrics addressed in the literature. In the **experimental study**, the questions are aimed at analyzing the performance of different textual representation models combined with effort estimation models. This approach ensures that the general objectives of the study are not confused with the specific objectives of each stage.

The first involves a Systematic Literature Mapping, through which the main studies, techniques, metrics, and approaches used in this domain are identified. Specific exploratory research questions guide this investigation and are detailed in the section 4.

The second consists of an Experimental Study that empirically evaluates models based on word embeddings, such as FastText and XLNet, combined with machine learning algorithms. The goal is to compare their performance, effectiveness, and computational cost in estimating effort, as outlined in the research questions and results presented in the section 6.

The findings of the systematic mapping highlight the relevance of the topic and the growing interest of the research community, as evidenced by the number of published studies. The analysis shows that NLP and machine learning techniques, especially deep learning models, tend to outperform traditional algorithms such as Random Forest and Support Vector Machines (SVM). It was also noted that many datasets used in the literature are derived from open-source projects and task management platforms such as Jira and Bamboo.

Systematic mapping contributes by providing a broad analysis of the most effective models and algorithms used in software effort estimation, offering insights into the current state of the art. Based on these findings, this study explores new techniques that may improve results and foster future research in the area.

The comparative experiment conducted in this work shows that both FastText and XLNet achieve similar results in estimating user stories. However, XLNet stands out for its ability to understand words in context, a critical advantage when dealing with complex textual requirements. This highlights the importance of contextual language models in scenarios where semantic understanding is essential.

In addition to these advancements, several factors must be considered for applying these techniques in real-world scenarios: (i) the availability of computational resources, as more sophisticated models require greater processing power; (ii) the length of user stories, since longer texts increase contextual complexity and demand more analytical capacity; and (iii) the language used in the application domain, which may impact the effectiveness of NLP models.

Machine learning and NLP techniques represent a promising approach for software effort estimation, helping reduce subjectivity and improve accuracy in the estimation process. By highlighting the challenges and opportunities in this field, this study provides a solid foundation for future work and practical applications.

The remainder of this paper is organized as follows: Section 2 presents the theoretical foundations. Section 3 reviews related work. Section 4 describes the systematic mapping. Section 5 details the development process using FastText and XLNet integrated into a deep learning model. Section 6 outlines the experimental design. Section 7 presents the results obtained. Section 8 discusses the development and evaluation of the application. Finally, Section 9 presents the conclusions of this work.

2 Background

2.1 Software Requirements

Software requirements refer to the specification of the functionalities that the system must meet and can be represented

in different ways, depending on the methodological approach adopted in the project. Among the main methodologies are agile and traditional [Pressman, 2011][Sommerville, 2011], both applied to software development, but with significant differences in their stages and forms of execution. In agile methodology, the main objective is to deliver results quickly, prioritize frequent deliveries to the customer, and reduce the complexity of documentation. In contrast, the traditional approach involves detailed and comprehensive documentation before starting the actual development. In the context of agile development, the user story technique is often used to accelerate the creation and management of requirements. This approach allows you to describe each requirement in the format of a simple narrative, facilitating understanding and speeding up the development process. On the other hand, traditional methodologies employ more formal techniques, such as use cases, which present requirements in a detailed and structured manner. This results in more comprehensive documentation compared to the practices adopted in agile development. Traditional approaches follow a structured and documented process, typically involving a series of steps for requirements gathering. These processes include requirement elicitation, analysis (using techniques like use cases, UML diagrams, and prototypes), and the documentation of requirements, adhering to strict documentation standards. [Sommerville, 2011][Wazlawick, 2010].

2.2 User Story

User stories are an approach used to specify requirements that describe the functional behavior of a software system. These requirements are organized hierarchically into four levels: Epic, Capabilities, Features, and User Stories. Each of these levels can be defined as follows: i) Epic: Represents a broad requirement, generally presented to customers or users, and usually covers an entire module of the system. Due to its complexity, it needs to be subdivided into smaller parts to enable development. ii) Capabilities: These refer to high-level behaviors that solve system problems. Each capability is broken down into several features to facilitate implementation. iii) Features: These are specific functionalities or system resources. Each feature has well-defined hypotheses or acceptance criteria and is subdivided into User Stories, which are assigned to development teams during iterations. iv) User Stories: Directed to the development team, they describe what a user wants to accomplish with the product and the reason for this need [Fávero *et al*., 2022]. The main purpose of a user story is to capture the essential elements of a [Cohn, 2005] software requirement. The most widely used format for describing a user story, as proposed by [Cohn, 2005], follows the structure: ”As a type of user, I want to accomplish something, for that goal or justification.” Table 1 illustrates an example user story using this pattern.

Table 1. example of user stories according to Cohn (2005).

US	As a seller, I would like to check the stock of a certain product to offer to a customer.

2.3 Software Effort Estimation

Effort estimation in software consists of calculating the work required to develop a specific requirement. Effort, in this context, is understood as a metric that quantifies the workload required to complete a task and can be expressed in units such as man/hour or man/day. In the case of software requirements, this metric can also consider the financial costs involved in development [Abdukalykov, 2011].

Several approaches are currently used to estimate effort in software requirements, which are classified as parametric/nonparametric algorithms and non-algorithmic methods. Parametric models use statistical and mathematical techniques to estimate effort, such as function point analysis [Choi *et al*., 2012] or the COCOMO II model [Boehm *et al*., 2000]. Non-algorithmic methods are based on machine learning algorithms, which perform estimates by learning from historical data from previous projects.

As pointed out by [Shepperd, 2007], effort estimation techniques can be grouped into three main categories: i) Human judgment-based techniques: These techniques are widely used in agile methodologies, and they rely on the subjective assessment of participants. However, they can lead to inaccurate estimates due to factors such as excessive optimism (e.g., the use of *planning poker*). ii) Parametric and algorithmic model-based techniques: represent a more objective alternative to human techniques, employing mathematical models or algorithms to perform the estimation (e.g., COCOMO and Function Points). iii) Machine learning-based predictive techniques: use algorithms such as linear regression, neural networks, and analogy-based methods. These approaches analyze historical data to make more accurate predictions about new projects.

Points per story represent a metric widely used in agile development methodologies, with the objective of expressing the amount of work necessary to develop a specific requirement, that is, a functionality of a [Cohn, 2005] system. One of the common approaches to assigning values to this metric is the Fibonacci sequence, which includes numbers such as 1, 2, 3, 5, 8, 13, 21, 34, 55, and so on.

2.4 Planning Poker

Today, most software projects perform effort estimates on requirements based on human judgment. This process usually relies on the developers’ experience and the development team’s collective perception. One of the most widely adopted techniques in this context is Planning Poker [Cohn, 2005; Grenning, 2002], a collaborative method inspired by the game of poker, which seeks to promote consensus among team members. In Planning Poker, each developer receives a set of numbered cards representing different levels of effort, known as story points. Each member chooses individually and without external influences for each requirement, a value that represents their estimate of the effort required for implementation. Then, all cards are revealed simultaneously, allowing the team to visualize the different perceptions of the complexity of the task. If there are significant discrepancies between the estimates, especially at the highest and lowest values, those responsible for these choices are encouraged to

justify their decisions. This process facilitates the exchange of knowledge among developers, leading to a more in-depth analysis of the technical challenges involved. After this discussion, a new round of voting can be held, or the team can choose to define a final value based on consensus. Although this approach is widely used, it presents challenges, such as cognitive bias, social influence, and variability between teams, which can impact the accuracy of estimates in a given context. Thus, recent research has explored automated methods based on artificial intelligence and machine learning, which seek to reduce subjectivity and improve the predictability of estimates [Kassem *et al*., 2023].

2.5 Natural Language Processing

Natural Language Processing, a subfield of computer science, is dedicated to developing techniques for understanding and interpreting human language. Its main objective is to create computational models capable of performing tasks such as automatic translation, textual analysis, information retrieval, and interactions between humans and machines [Russell, 2016].

Among the main applications of NLP, text analysis and interpretation stand out, especially due to the vast amount of textual data available on the Internet. These analyses have been improved with the use of machine learning (ML) techniques, which, by extracting specific characteristics from texts, enable computational models to perform advanced tasks, such as text classification and other analyses [Hirschberg and Manning, 2015].

For NLP-based models to be able to manipulate and interpret texts, it is essential to subject them to a training process. This training requires a substantial set of texts related to the context of interest, known as a corpus, which allows the model to learn and perform tasks aligned with the domain being analyzed [Cambria and White, 2014].

According to [Sinclair and Wynne, 2004], a corpus is defined as a set of texts stored in electronic format. To achieve efficient performance, NLP models need to access a significant amount of texts belonging to the specific domain and in the language in which they will be applied. However, one of the limitations of NLP is that many models are developed only for languages with extensive support for textual data, such as English, French, Spanish, German and Chinese [Hirschberg and Manning, 2015].

In order for machine learning algorithms to understand texts, it is necessary to convert them into a format suitable for computational interpretation. This conversion occurs during the preprocessing stage in text mining tasks, which ensures a more efficient representation of the contents.

2.6 Textual Representation

Textual representation plays a crucial role in text mining. Various techniques have been developed to capture textual features in the most effective way, aiming, for instance, to identify patterns and similarities between text elements. In the following, we present the word embeddings model, which served as the foundation for the textual representation techniques adopted in this study.

2.7 Word Embeddings

Word embeddings models are a reference in the field of Natural Language Processing (NLP) for textual representation, and are considered a significant advancement over classical methods, particularly due to their learning approach and the way words are represented [Naseem *et al*., 2021]. These models stand out for their ability to learn representations automatically by using neural networks to extract features, mapping words into a space where those with similar meanings are positioned closer together [Bengio *et al*., 2013; Jurafsky and Martin, 2019].

Compared to classical models—which often represent words using thousands or even millions of dimensions—word embeddings typically use vectors with only dozens or hundreds of dimensions to represent a word [Goldberg, 2022], making processing and mapping more efficient. Furthermore, this mapping allows the model to capture semantic relationships (for instance, between an infinitive verb and its conjugated forms), enabling words to be represented by different numerical vectors according to their semantic meaning [Mikolov *et al*., 2013a; Zhang *et al*., 2015]. **Figure 1** illustrates an example of word mapping and the relationships between word representations.

Once words are represented as vectors of real numbers, they can be positioned in an n-dimensional space, allowing the calculation of distances between them. This enables the measurement of their semantic proximity — the closer the vectors, the more similar the meanings; conversely, greater distances reflect semantic divergence, as illustrated in **Figure 1** [Goldberg, 2022].

According to Haj-Yahia et al. [Haj-Yahia *et al*., 2019], the predictive methods used for generating word embeddings can be broadly classified into two categories: (i) context-free and (ii) contextualized models. The following sections explain the concepts and structural distinctions between these two types.

2.8 Context-Free and Contextualized Word Embeddings

Traditional word embeddings, often referred to as context-free embeddings, map each word to a fixed vector regardless of the sentence in which it appears. This approach presents limitations, especially for polysemous words, as it fails to capture variations in meaning depending on context [Jurafsky and Martin, 2019]. Additionally, these models are typically unidirectional, they process input text from left to right, which prevents them from fully understanding the semantics of a complete sentence. Another limitation is their shallow architecture, as they represent words using only a single layer [Mikolov *et al*., 2013a].

These drawbacks can be mitigated by contextualized embeddings, which adapt a word's representation based on the surrounding words in the sentence. As a result, different contexts yield different vectors for the same word, addressing issues such as polysemy and ambiguity [Jurafsky and Martin, 2019].

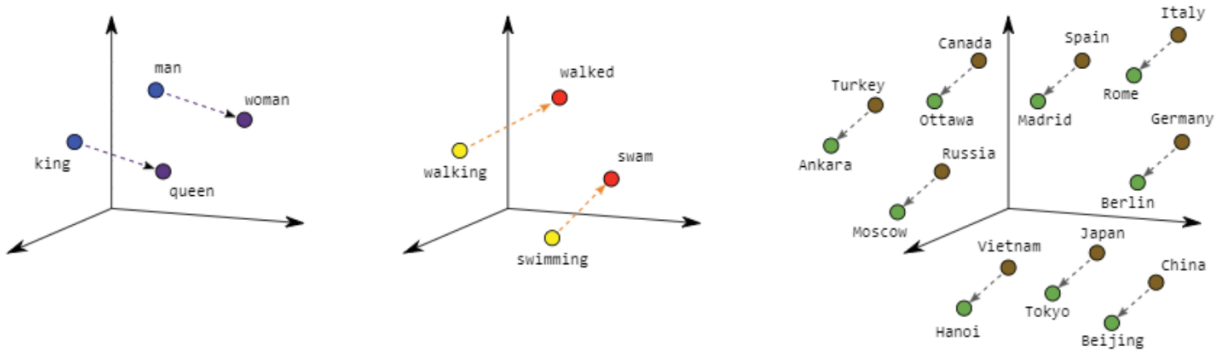

Figure 1. Vector representation of word embeddings [Mikolov *et al*., 2013b].

2.9 BERT and Transformer Models

Among the most significant advances in contextualized word embeddings is the Bidirectional Encoder Representations from Transformers (BERT), introduced by Devlin et al. [Devlin *et al*., 2018]. BERT leverages the Transformer architecture [Vaswani *et al*., 2017], which is based entirely on self-attention mechanisms, enabling the model to capture complex dependencies across entire sequences. Unlike traditional models that process text in a unidirectional manner, BERT is pre-trained using a masked language modeling (MLM) objective, allowing it to jointly condition on both left and right contexts in all layers.

The introduction of BERT marked a turning point in NLP, setting new benchmarks across a wide range of tasks such as question answering, sentiment analysis, and text classification. Its architecture has inspired the development of several other powerful models, including RoBERTa, ALBERT, and XLNet, which introduce improvements in training strategies and model efficiency.

These Transformer-based models demonstrate superior performance by learning deep contextualized representations, significantly advancing the capabilities of NLP systems in understanding semantics, handling ambiguity, and modeling long-range dependencies in text.

2.10 FastText

FastText is an embedding model developed as an open-source API (Application Programming Interface) by Facebook. Similar to other widely used models such as Word2Vec and GloVe [Pennington *et al*., 2014], FastText aims to transform textual features into vector representations (embeddings), facilitating their use in machine learning models. One of its main advantages is its speed in learning word representations [Bojanowski *et al*., 2017], which is largely attributed to its ability to capture the internal structure and morphology of words independently [Santos *et al*., 2017].

The model adopts the skip-gram architecture, a training objective in which the system attempts to predict surrounding context words given a target word. This allows the model to learn how words co-occur within a defined context window in a corpus [Mikolov *et al*., 2013a]. FastText, as an extension of Word2Vec, is designed to work directly with raw texts. Its vector representations are constructed by combining character n-grams, which enables the model to incorporate subword information. This makes it particularly effective in interpreting rare, incomplete, or misspelled words, as well as capturing semantic information based on the sequence and organization of characters. Moreover, the model considers the context of adjacent words on both the left and right sides, enhancing its ability to understand meaning in local contexts.

FastText was selected for textual feature extraction in this study due to its various advantages, including low computational cost and robustness to lexical variations. Its use aims to evaluate its performance in the proposed experiments, emphasizing its efficiency and suitability for the task [Ghosal and Jain, 2023].

2.11 XLNET

XLNet is a generalized autoregressive model widely used in text classification tasks. Developed as a combination of concepts from ELMO and BERT models [Devlin *et al*., 2018], XLNet is capable of capturing the full context of a piece of information, considering both sides of the sentence. The model operates in two main phases: pre-training and fine-tuning. During pre-training, XLNet rearranges the words of the sentences to understand and learn the semantic meaning in a bidirectional manner, overcoming the limitations of traditional linguistic models that perform one-dimensional analysis.

XLNet's main approach involves extracting textual features by reconstructing the input texts through permutations and combinations, which allows it to explore the text in its entirety and perform bidirectional predictions. To achieve this, the model is structured with a self-attention layer that operates in two streams, as illustrated in **Figure 2**.

As illustrated in **Figure 2**, the calculation flow of the content is shown on the left side, while the *query* is shown below. In the calculation process, if the goal is to predict x_1 based only on its position, the representation will be KV = [h_1,h_2,h_3], and Q = g_1. If the goal is to predict the surrounding tokens, the representation will be KV = [h_1,h_2,h_3, h_4] and Q = h_1. On the right side, the calculation of the outputs is shown, where the weights of h and g are applied to (x_i) and w. From there, a mask is used for both the content and the *query* to calculate the outputs [Wang and Zhang, 2022].

XLNet was used in this study due to its ability to capture contextual information bidirectionally, which aligns directly with the format of textual requirements, where it is essential

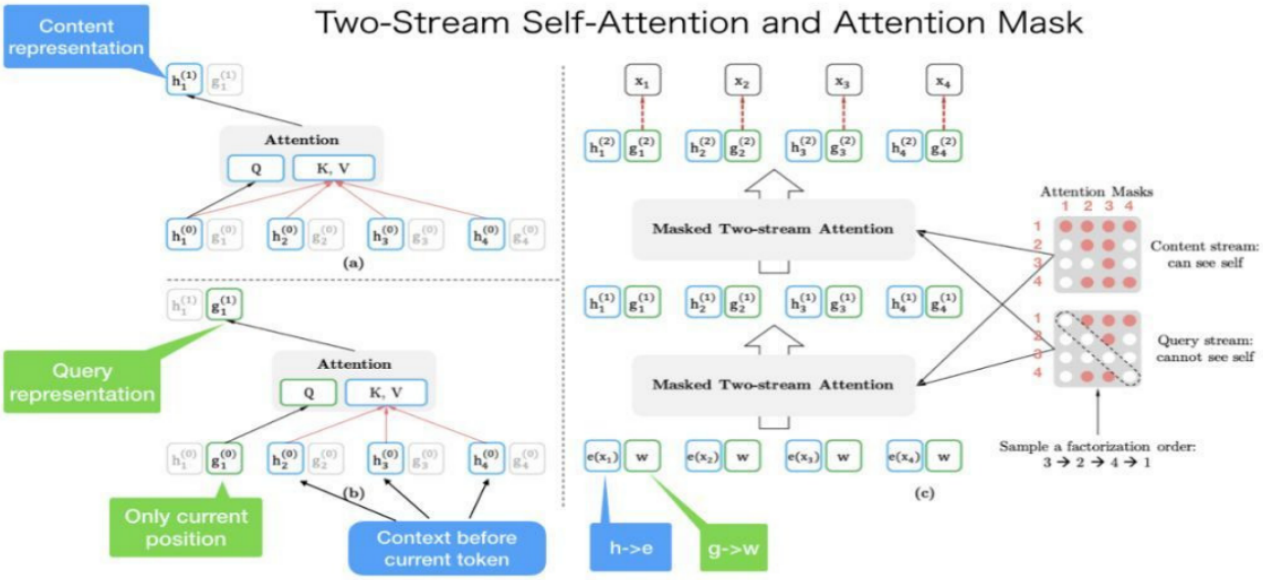

Figure 2. Representation of the XLNet model [Wang and Zhang, 2022].

to ensure the accuracy of semantic representations of user stories. Unlike models such as FastText, XLNet understands complex dependencies between words, enhancing prediction accuracy when performing effort estimation. Several studies have demonstrated its effectiveness in natural language processing tasks [Wang and Zhang, 2022], supporting its suitability for the present study.

2.12 Deep Learning

Deep Learning (DL) is a type of neural network that has gained prominence in various fields. Some algorithms in this subfield, such as Recurrent Neural Networks (RNN) and Convolutional Neural Networks (CNN), are widely used in text classification tasks.

One of the limitations encountered in RNNs is the vanishing gradient problem, also known as the disappearing gradient. This issue arises during the training of deep neural networks using backpropagation algorithms once the gradients of the weights in the deeper layers of the network become small as training progresses. The use of backpropagation algorithms results in insignificant updates to the weights of these layers, making the model less efficient than desired.

To deal with this problem, the LSTM (*Long Short-Term Memory*) deep learning model was developed, which stands out from RNNs for its ability to maintain long-term memory. The main difference between this model and conventional RNNs lies in the separation between the output of each node and its memory and the existence of self-recurrent *feedback* connections with a delay of one step. This allows LSTMs to overcome the *vanish gradient* problem and capture long-term dependencies in data sequences, making them more effective in different ML and NLP tasks [Van Houdt *et al*., 2020].

2.13 Machine Learning Models for Effort Estimation

The present research aims to develop a model to estimate the effort in textual requirements. For this, NLP techniques combined with deep learning models are used. Text extraction and analysis play a key role in the accuracy of the model, ensuring the representation of words through embedding vectors.

In this research, the XLNet and FastText models were used, which have distinct characteristics: the first is a contextualized model, while the second does not consider the context of the words. These textual representations are processed by deep neural networks to identify patterns and make more accurate predictions of the effort required for software development.

3 Related Works

Several studies have investigated models and techniques to improve the accuracy of effort estimation based on textual requirements.

Choetkiertikul *et al*. [2018] proposed the Deep-SE model, composed of an LSTM and Recurrence Highway Network (RHN), trained on 23,313 issues from 16 open-source projects. Although the model achieved substantial improvements over traditional baselines, it relied on non-contextual embeddings and lacked domain adaptation.

Gultekin and Kalipsiz [2020] expanded estimation to the sprint level using Gradient Boosting Regression (GBR) on large open-source datasets. Although effective for aggregated predictions, their approach did not explore deep learning or contextualized embeddings, which are capable of modeling nuanced semantic patterns.

Ramessur and Nagowah [2021] proposed a regression-based model incorporating contextual variables and extensive preprocessing. However, the study did not employ deep learning methods, limiting the ability to capture complex

relationships within user story texts.

More recently, Fávero *et al*. [2022] integrated BERT embeddings into Deep-SE and evaluated the model on a unified repository of software projects. Their results demonstrated that contextualized transformer embeddings can substantially improve accuracy, although only a single contextual model was evaluated.

Kassem *et al*. [2023] proposed a Hierarchical Attention Network to capture document-level structure, achieving notable improvements in MAE and MdAE. Nonetheless, the study was restricted to a single dataset and did not compare multiple embedding strategies, limiting its generalizability.

Although these works have advanced effort estimation, important limitations remain. As summarized in Table 2, most studies rely on general pre-trained models without domain-specific fine-tuning, evaluate only a single textual representation, or use restricted datasets. Moreover, existing work rarely compares contextualized and non-contextualized models under a unified experimental design.

In parallel, Fávero *et al*. [2018] conducted a systematic mapping focused exclusively on Estimation by Analogy (EbA) techniques between 2007 and 2017. While the study identified historical trends, it did not address modern machine learning or deep learning models applied to textual requirements.

No other surveys or systematic mappings were found that specifically address machine learning for effort estimation based on user story text, which further highlights the relevance of the present study.

The present study contributes to the field by evaluating multiple modern language models, FastText and XLNet, within a unified deep learning architecture inspired by Deep-SE, which incorporates domain-specific unsupervised fine-tuning and integrates systematic mapping with empirical experimentation.

4 Systematic Mapping

This Systematic Mapping was conducted following the guidelines proposed by [Petersen *et al*., 2015]. Additionally, the systematic literature review guidelines by [Kitchenham and Charters, 2007] were used when complementary, given the similarities in the processes of identifying and selecting primary studies. The SMS followed five main steps [Petersen *et al*., 2008], which are described in the following subsections.

4.1 Research Questions (RQs)

This systematic mapping aims to identify studies that apply machine learning techniques to perform *software* effort estimation on requirements. **Table 3** presents the RQs defined to guide this SMS.

4.2 Search String

Our research questions guided the identification of the main search, which included “requirement estimation” and “machine learning”. We then extracted synonyms and alternative expressions for each term to build tailored search strings for each database. The derived alternatives for both key terms are summarized in **Table 4**.

These keywords were chosen mainly because they emphasize the theme of systematic mapping, and they are present in several related studies, for exemple [Choetkiertikul *et al*., 2018], [Fávero *et al*., 2022] and [Kassem *et al*., 2023]. We used boolean operators to link the key terms and their synonyms when connecting the search string. As a result, we formed the following search string:

(“requirements estimation” OR “user story estimation” OR “story point estimation” OR “estimating story points”) AND (“machine learning” OR “deep learning” OR “neural networks”)

The following online databases were searched: IEEE Xplorer[1], ACM Digital Library[2] and Scopus[3]. The choice to use these databases was due to the *Scopus* base being the largest database of titles and abstracts [Keele *et al*., 2007], and the others being popular digital libraries used in related studies, as they are the primary search sources, in addition to being the most used digital libraries for systematic studies [Britto *et al*., 2014], [Costa and Salvador, 2015] and [Hussain *et al*., 2013].

Each was evaluated article based on its title and abstract and performed an inclusion criteria (IC) and exclusion criteria (EC) assessment as shown in **Table 5**.

In addition to the string-based search, was adopted a hybrid search strategy that included backward and forward snowballing, as recommended by [Wohlin, 2014]. This ensured the inclusion of relevant studies potentially missed by the initial query-based search.

4.3 Selection Process

The study was carried outthrough a systematic search in three electronic databases, following a selection process based on predefined criteria. **Figure 3** illustrates the main steps of this process.

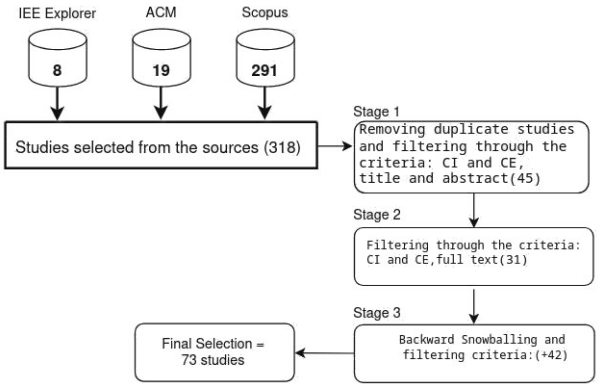

Figure 3. Search and selection SM process

”In cases of duplicate publications, was considered the most recent version. Using the search string in the Scopus database, was recovered 291 studies. In total, the search

[1]https://ieeexplore.ieee.org/
[2]https://dl.acm.org/
[3]https://www.scopus.com

Table 2. Comparison of related works, identified gaps, and contributions of the present study

Author / Year	Technique / Model	Dataset	Main Limitations	Gaps Addressed by This Study
Choetkiertikul et al. (2018)	Deep-SE (LSTM + RHN)	23k OSS issues	Non-contextual embeddings; single architecture; no domain adaptation.	Adds contextual (XLNet) vs non-contextual (FastText) comparison and domain-specific fine-tuning.
Gultekin and Kalipsiz (2020)	GBR (Traditional ML)	OSS sprint datasets	No deep learning; limited semantic modeling; no embedding comparison.	Introduces deep models and evaluates multiple textual representations.
Ramessur and Nagowah (2021)	Regression ML	Private dataset	No DL; shallow features; no contextual modeling.	Uses deep neural architecture with contextual embeddings and fine-tuning.
Fávero et al. (2022)	BERT + Deep-SE	Unified OSS repository	Single contextual model; no contrast with non-contextual baselines.	Performs multi-embedding evaluation and integrates mapping + experimentation.
Kassem et al. (2023)	HAN (Attention)	Single OSS dataset	Dataset-specific; no multi-model comparison; no domain tuning.	Evaluates two distinct models; tests generalization; applies domain adaptation.
Fávero et al. (2018)	Systematic Mapping	Multiple datasets	Focus on Estimation by Analogy (EbA); no ML/DL for textual estimation.	Expands scope with ML/DL models and updates systematic evidence.

Table 3. Research Questions

ID	Questions	Rationale
RQ1	How many publications have performed software effort estimation on requirements using machine learning?	Identify the number of studies developed over the years
RQ2	Which datasets used in the selected studies?	Identify the data sets used in the studies and how frequently they are used will be analyzed.
RQ3	What machine learning algorithms were used in these studies?	Identify and verify which algorithms have been used in the context of effort estimation in *software* requirements.
RQ4	What evaluation metrics were used in these studies?	Identify which metrics were used in the analyzed studies, highlighting how frequently they are used.

Table 4. List of keywords and their synonyms

Keywords	Synonyms
requirement estimation	user story estimation, story point estimation, estimating story points.
machine learning	deep learning, neural networks

Table 5. List of inclusion and exclusion criteria

Inclusion Criteria
IC1 - The study concentrates estimation in requirements, answers; **IC2** - The study contributes to at least one of the research questions; and **IC3** - The study was published between 2005 and 2023.
Exclusion Criteria
EC1 - Abstract or extended abstract without full text **EC2** - The manuscript does not have sufficient bibliographical information, for instance, the publisher **EC3** - Study is not written in English; **EC4** - Study is a copy or an older version of another publication already considered. In these cases, the most current version is considered **EC5** - No access to the full study **EC6** - Study is not a primary Study.

returned 318 publications: 8 from IEEE, 19 from the ACM Digital Library, and 291 from Scopus. Below, it describes how each stage of the selection was carried out.

Stage 1: In the 1st stage, was removed duplicate studies and applied the selection criteria on title, abstract, and keywords, resulting in 45 selected studies.

Stage 2: In the 2nd stage, the selection criteria were applied considering the full text, resulting in a set of 31 studies.

Stage 3: Over these 31 studies that remained in 2nd stage, we performed backward snowballing in 3rd stage. In the snowballing process, we looked at all the references from the 31 studies selected and was applied the selection criteria in the title, abstract, and keywords, leading to 54 new studies. The selection criteria were again applied to the 54 studies considering the full-text analysis, resulting in 42 studies.

As a result, we had 73 studies to analyze (31 from the sources and 42 from backward snowballing). **Figure 3**summarizes the stages and their results, showing the progressive reduction of the number of studies throughout the selection stages.

4.4 Data Collection and Analysis

There were several meetings between the authors to decide on what information to extract from each of the selected studies. A shared Excel document[4] was created, and each paper was given a unique ID. Was extracted publication details that included: publication venue, year, country, type of publication (conference, workshop, or journal), and authors. We further extracted the title, algorithms, metrics, and databases used. Next, we listed the publication date for each paper to determine if there was a trend in the number of publications per year on related research.

4.5 Results Analysis

Based on the previously defined research questions, this section presents the results of the systematic mapping study. The analysis focuses on the selected studies, with emphasis on the datasets and techniques used, in order to answer the proposed questions.

*RQ*1**: How many publications have performed software effort estimation on requirements using machine learning?** This research question aims to analyze the number of studies related to the topic in order to demonstrate its relevance and how it has evolved over time.

Figure 4 presents a graph showing the distribution of publications from 2005 to 2023, all of which address effort estimation for software requirements using machine learning techniques.

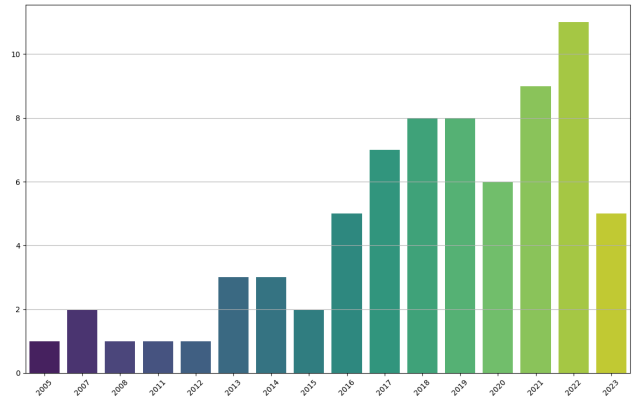

Figure 4. Number of studies published between 2005 and 2023.

The graph reveals a clear upward trend in the number of published studies over the years, indicating growing academic interest and the potential for continued research in the field of software effort estimation using machine learning. This progression suggests that the topic is gaining increasing relevance and visibility within the software engineering research community.

Additionally, **Figure 5** presents the distribution of publication types among the selected studies, categorized as journal articles, conference papers, and book chapters. This classification highlights the diversity of publication venues and the multidisciplinary nature of the research.

[4]An abbreviated version of this form is available at `https://docs.google.com/spreadsheets/d/1-2nJd-I_mmJw2sA5stIdlLScBnwnusnX/edit?usp=sharing&ouid=108975222690704602861&rtpof=true&sd=tru`

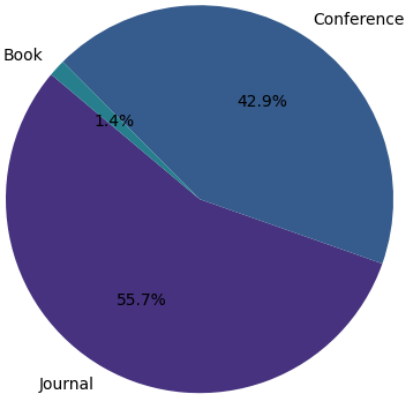

Figure 5. Types of publication.

The analysis shows that the majority of the selected studies were published in journals (55.7%), followed by conference proceedings (42.9%), and a small portion in books (1.4%). This distribution reflects a strong preference for disseminating research findings through peer-reviewed journals and conferences, which are the primary channels for scientific communication in the area.

Figure 6 displays the distribution of studies by country, aiming to identify the nations that have contributed most to the research domain. The data reveal that India (29.04%) and the United States (11.8%) are the leading contributors in terms of publications related to software requirements effort estimation using machine learning. This suggests a strong research interest in these countries, possibly driven by academic, industrial, or governmental initiatives.

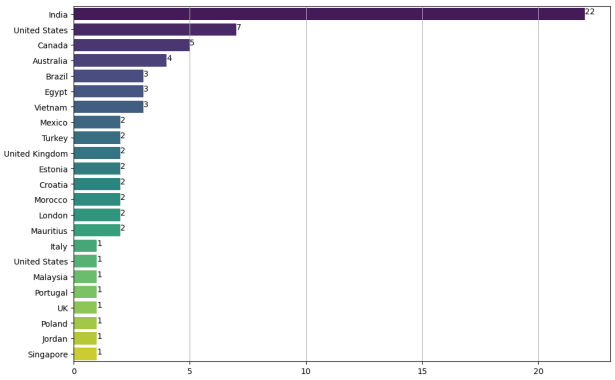

Figure 6. Distribution of studies by country.

*RQ*2**: What are the datasets used in the selected studies?**

In response to RQ2, the analysis of the selected articles revealed the recurring use of several datasets across different studies. Most of these datasets are publicly available and are commonly used in empirical research on software effort estimation. Since the focus is on the application of machine learning techniques, training models typically involve data from various types of software projects. **Table 6** provides a brief description of each identified dataset.

It is essential to highlight that open-source projects typically use different criteria for effort estimation than commercial projects, which tend to evaluate effort and time based on

Table 6. Description of datasets used in the selected studies.

Dataset	Description	Studies that used the dataset
Zia et al.	Dataset used for requirement-based effort estimation using NLP techniques.	[Sánchez *et al*., 2022], [Rodríguez Sánchez *et al*., 2023], [Ramessur and Nagowah, 2021], [Choetkiertikul *et al*., 2018], [Scott and Pfahl, 2018], [Satapathy and Rath, 2017] [Bilgaiyan *et al*., 2019], [Prasada Rao *et al*., 2018], [Khuat and Le, 2016],[Kaushik *et al*., 2020], [Sharma and Chaudhary, 2020], [Hemrajani and N, 2021], [Arora *et al*., 2021], [Khuat and Le, 2016], [Bilgaiyan *et al*., 2019], [Khuat and Le, 2018], [Khan and Qureshi, 2014]
Choetkiertikul et al.	Real project data for training ML models in effort prediction.	[Tawosi *et al*., 2022] [Phan and Jannesari, 2022b] [Kumar *et al*., 2022] [Gupta and Mahapatra, 2022] [Kassem *et al*., 2023] [Fu and Tantithamthavorn, 2022] [Abadeer and Sabetzadeh, 2021] [Turic *et al*., 2023] [de Morais, 2021] [Kanmani *et al*., 2007] [Braga *et al*., 2007]
Tawos	Includes user stories and estimated effort annotations.	[Tawosi *et al*., 2022]
Private	Proprietary datasets from internal company projects.	[Sarro *et al*., 2020], [Zakrani *et al*., 2018], [Gultekin and Kalipsiz, 2020], [Soares, 2018], [Scott and Pfahl, 2018], [Najm *et al*., 2019], [Moharreri *et al*., 2016], [Choetkiertikul *et al*., 2018] [Sudarmaningtyas and Mohamed, 2021]
COCOMO	Classic dataset based on Boehm's cost estimation model.	[Kanmani *et al*., 2007] [Hussein *et al*., 2017] [Nassif *et al*., 2013] [Araujo *et al*., 2017] [Azzeh *et al*., 2015]
NASA	Historical project data from NASA, often using COCOMO.	[Praynlin and Latha, 2013],[Kultur *et al*., 2008], [Yousef *et al*., 2017], [Khuat and Le, 2018], [Minku and Yao, 2013]
China	Dataset from Chinese software projects.	[Hussein *et al*., 2017], [Nassif *et al*., 2013]
Jira	Extracted from Jira systems: user stories, tasks, and logged effort.	[Malgonde and Chari, 2019], [Zakrani *et al*., 2018]
Open source projects	Various datasets from OSS repositories with effort data.	[Arora *et al*., 2021],[Malgonde and Chari, 2019], [Porru *et al*., 2016], [Abrahamsson *et al*., 2011], [Suresh Kumar *et al*., 2022]
Desharnais	Contains function points and productivity metrics.	[Abnane *et al*., 2019], [Nassif *et al*., 2013], [Srinivasan and Fisher, 1995], [Singh and Kumar, 2020], [Araujo *et al*., 2017], [Azzeh *et al*., 2015]
Maxwell	Projects from the telecom domain with effort info.	[Abnane *et al*., 2019], [Nassif *et al*., 2013]
PROMISE	Public repository with SE datasets, including effort data.	[Kumar *et al*., 2020],[Kaushik and Singal, 2022], [Hamouda, 2014]
Albrecht	Early dataset using Function Points for estimation.	[Nassif *et al*., 2013], [Araujo *et al*., 2017]
Kemerer	Size and effort data from business applications.	[Nassif *et al*., 2013],[Najm *et al*., 2019] , [Araujo *et al*., 2017], [Azzeh *et al*., 2015]
Kitchenham	Dataset from systematic reviews on cost estimation.	[Nassif *et al*., 2013]
ISBSG	Industrial repository with anonymized project data.	[Srinivasan and Fisher, 1995], [Pospieszny *et al*., 2018], [Hamouda, 2014]
Bamboo	Data from CI/CD pipelines, including deployment effort.	[Marapelli *et al*., 2020]

delivery deadlines and contractual commitments. As a result, using diverse datasets during model training is essential to improving the generalization and applicability of the effort estimation models, regardless of the project context (**Figure 7**). However, this practice,training with heterogeneous datasets, was not commonly observed in the reviewed studies.

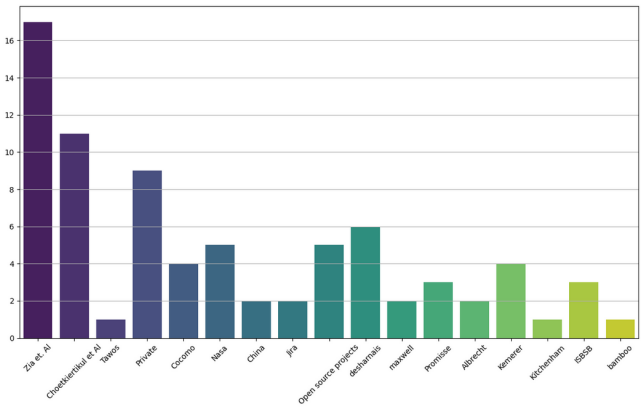

Figure 7. Datasets used

*RQ*3**: What machine learning algorithms were used in these studies?**

The analysis of the selected studies revealed the use of various machine learning algorithms for effort estimation. To identify which algorithms are most commonly adopted in this context, **Table 7** provides a summary that lists each study alongside the specific algorithms applied. This overview allows for a comparative understanding of the algorithmic preferences and trends within the research community.

Figure 8 presents a summary of the main algorithms used in the analyzed works. Several studies have employed multiple algorithms to perform the same estimation, with the aim of investigating different approaches and identifying those that offer the best results. An example of this is the study by [Malgonde and Chari, 2019], which applies machine learning algorithms to predict the effort required in agile software development.

The analysis of the studies revealed that the most commonly adopted machine learning algorithms for software requirements effort estimation were Artificial Neural Networks (ANN), Support Vector Machines (SVM), and decision tree–based methods such as Random Forest and AdaBoost. Among them, ANN/NN stood out as the most frequently used, appearing in 20 studies, followed by SVM and Random Forest, each cited in 10 studies. This pattern suggests a clear preference within the research community for these models, likely due to their effectiveness in capturing complex relationships within the data. In contrast, more recent algorithms such as XGB, CatBoost, and GRB were seldom explored, indicating opportunities for further investigation."

*RQ*4**: What evaluation metrics were used in these studies?** In response to RQ4, several studies also reported concerns regarding the evaluation process, particularly the choice and application of performance metrics. These metrics play

Table 7. Use of algorithms in effort estimation of software requirements

Algorithm	Study
KNN	[Sánchez *et al*., 2022], [Malgonde and Chari, 2019], [Sarro *et al*., 2020], [Malgonde and Chari, 2019], [Suresh Kumar *et al*., 2022] [Abnane *et al*., 2019]
AR	[Rodríguez Sánchez *et al*., 2023], [Ramchurreetoo and Hurbungs, 2022], [Ramessur and Nagowah, 2021], [Arora *et al*., 2021], [Malgonde and Chari, 2019], [Satapathy and Rath, 2017], [AG *et al*., 2021], [Nassif *et al*., 2013], [Suresh Kumar *et al*., 2022], [Moharreri *et al*., 2016]
Random Forest	[Rodríguez Sánchez *et al*., 2023], [Sarro *et al*., 2020], [Arora *et al*., 2021], [Satapathy and Rath, 2017], [AG *et al*., 2021], [Singh and Kumar, 2020], [Moharreri *et al*., 2016], [Sree *et al*., 2017], [Suresh Kumar *et al*., 2022], [de Morais, 2021]
AdaBoost	[Rodríguez Sánchez *et al*., 2023], [Arora *et al*., 2021], [Suresh Kumar *et al*., 2022]
SVM	[Tawosi *et al*., 2022], [Ramchurreetoo and Hurbungs, 2022], [Ramessur and Nagowah, 2021], [Malgonde and Chari, 2019], [Soares, 2018], [Abrahamsson *et al*., 2011], [Scott and Pfahl, 2018], [Hidmi and Sakar, 2017], [AG *et al*., 2021], [Pospieszny *et al*., 2018]
ANN / NN	[Suresh, 2022], [Gupta and Mahapatra, 2022], [Phan and Jannesari, 2022a], [Ramessur and Nagowah, 2021], [Malgonde and Chari, 2019], [Khuat and Le, 2016], [Yousef *et al*., 2017], [Kumar *et al*., 2020], [AG *et al*., 2021], [Bilgaiyan *et al*., 2019], [Sree *et al*., 2017], [Dhir *et al*., 2017], [Praynlin and Latha, 2013], [Kultur *et al*., 2008], [Azzeh *et al*., 2018], [Azzeh *et al*., 2015], [Kanmani *et al*., 2007], [Kaushik and Singal, 2022], [Khan and Qureshi, 2014], [Kocaguneli *et al*., 2011]
NB	[Sarro *et al*., 2020], [Ramchurreetoo and Hurbungs, 2022], [Moharreri *et al*., 2016]
GNN	[Phan and Jannesari, 2022b]
RHN	[Fávero *et al*., 2022], [Abadeer and Sabetzadeh, 2021], [Choetkiertikul *et al*., 2018]
DBN	[Kaushik *et al*., 2020], [Gupta and Mahapatra, 2022]
RNN	[Kassem *et al*., 2023], [Fu and Tantithamthavorn, 2022], [Marapelli *et al*., 2020], [Panda *et al*., 2015], [Hussein *et al*., 2017], [de Morais, 2021]
MLP	[Fu and Tantithamthavorn, 2022], [Gultekin and Kalipsiz, 2020], [Singh and Kumar, 2020], [Pospieszny *et al*., 2018], [Minku and Yao, 2013], [Hamouda, 2014], [Araujo *et al*., 2017], [Braga *et al*., 2007]
RL	[Ramessur and Nagowah, 2021], [Arora *et al*., 2021], [Abrahamsson *et al*., 2011], [Sharma and Chaudhary, 2020], [Hemrajani and N, 2021], [Fedotova *et al*., 2013], [Singh and Kumar, 2020], [Kocaguneli *et al*., 2011]
LSTM	[Fávero *et al*., 2022], [Choetkiertikul *et al*., 2018], [Abadeer and Sabetzadeh, 2021], [Marapelli *et al*., 2020], [de Morais, 2021]
XGB	[Arora *et al*., 2021],
CatBoost	[Arora *et al*., 2021]
BN	[Malgonde and Chari, 2019], [Porru *et al*., 2016], [Turic *et al*., 2023], [Dragicevic *et al*., 2017]
GRNN	[Panda *et al*., 2015], [Prasada Rao *et al*., 2018]
SVR	[Zakrani *et al*., 2018],
GRB	[Suresh Kumar *et al*., 2022],

a crucial role, as they directly influence both the interpretation of results and the selection of the most suitable machine learning technique.

Figure 9 summarizes the evaluation metrics used across the analyzed studies. The most common metrics include: Accuracy, Mean Square Error (MSE), Mean Relative Error (MRE), Variance, Coefficient of Determination (R^2), Mean Magnitude of Relative Error (MMRE), Root Mean Square Error (RMSE), and Percentage of Estimates within a threshold (PRED). Other metrics also identified include Median Absolute Error (MdAE), Standardized Accuracy (SA), R^2 Score, Mean Balanced Error (MBE), Median Magnitude of Relative Error (MdMRE), SPPercentage, Precision, Recall, F-Measure, Magnitude of Relative Error (MER), Standard Deviation (SD), Receiver Operating Characteristic (ROC) Curve, Relative Root Square Error (RRSE), Relative Absolute Error (RAE), and Mean Absolute Error (MAE).

The diversity of metrics observed highlights the lack of standardization in evaluation approaches, reinforcing the need for greater consistency in future studies to enable more reliable comparisons between models.

Observing the results, it is possible to conclude that the three most used metrics are: MAE (15.5%), MRE (15.5%), and MMRE (12.5%). Furthermore, it is important to mention that different metrics are used to evaluate the performance of a single model and that it would be difficult for a model to be evaluated using just one evaluation metric.

4.6 Discussion of Systematic Mapping

This section provides a critical analysis of the findings obtained from the research questions RQ1 through RQ4. The goal is to identify patterns, gaps, and opportunities in the literature on effort estimation based on requirements using machine learning techniques, thereby supporting the relevance and scope of the present study.

RQ1: How many publications have performed software effort estimation on requirements using machine learning? The gradual increase in the number of publications over the years demonstrates a growing interest in effort estimation through machine learning. However, the concentration of studies in specific regions (mainly North America, Europe, and Asia) reveals a lack of geographical diversity in contributions. This concentration has implications for the generalizability of findings, since most of the datasets and case studies originate from contexts with specific development practices, team structures, and project management approaches. The relative absence of studies from regions such as Latin America and Africa also limits the exploration of challenges faced in emerging software industries, where resources, cultural factors, and development methodologies may differ substantially. Consequently, although the field shows signs of growth, it remains geographically uneven and potentially biased toward dominant research ecosystems, highlighting the need for broader participation from underrepresented communities to achieve more comprehensive and inclusive progress.

RQ2: What are the datasets used in the selected studies? The majority of studies analyzed rely on datasets from open-source projects, likely due to their public availability and standardized format. While this facilitates reproducibility and access, it limits the variety of contexts being assessed—particularly commercial environments. Although addressing dataset diversity is beyond the scope of this study, this issue is acknowledged as a promising direction for future research. The reliance on open-source datasets, while beneficial for comparability and reproducibility, raises critical concerns about external validity. Industrial projects differ significantly from open-source initiatives in terms of requirements complexity, development practices, and team dynamics. As a result, models validated solely on open-source data may fail

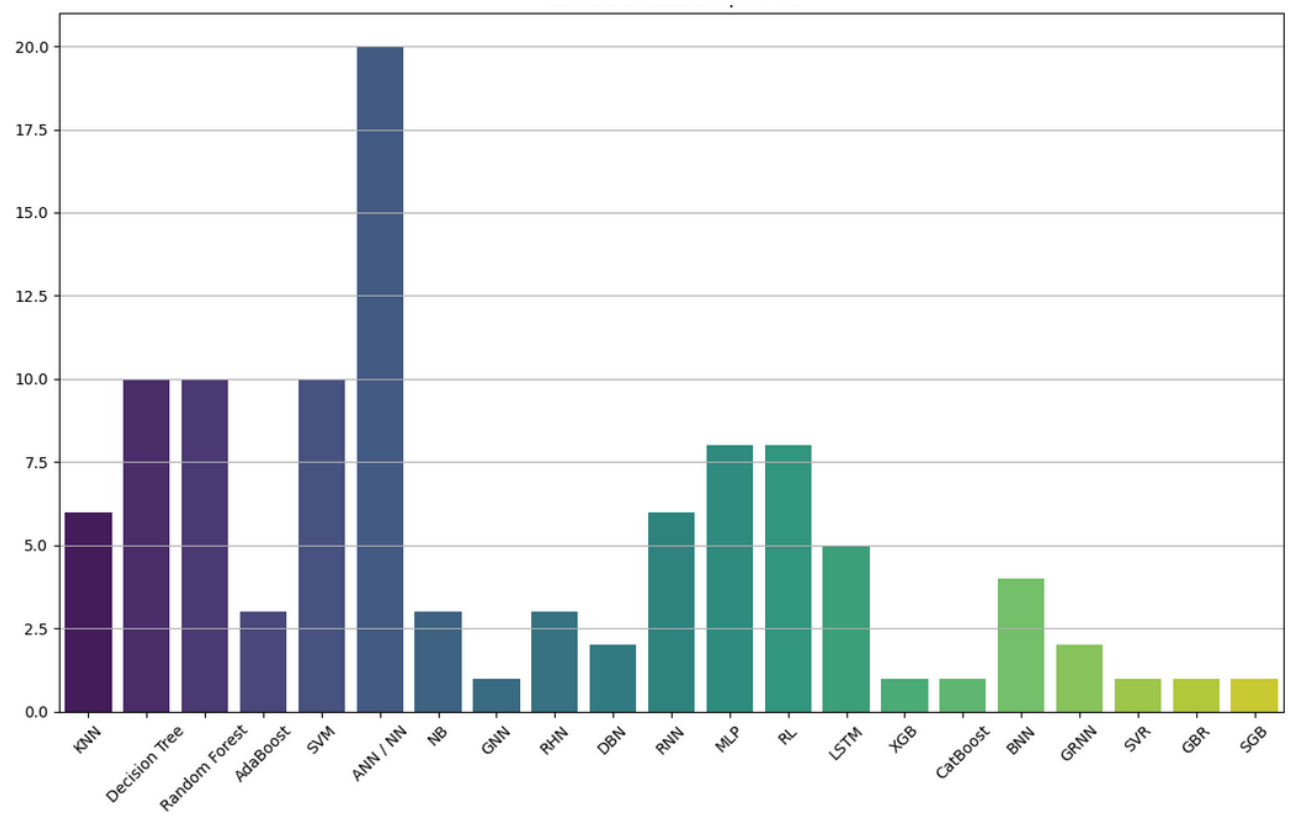

Figure 8. Algorithms used.

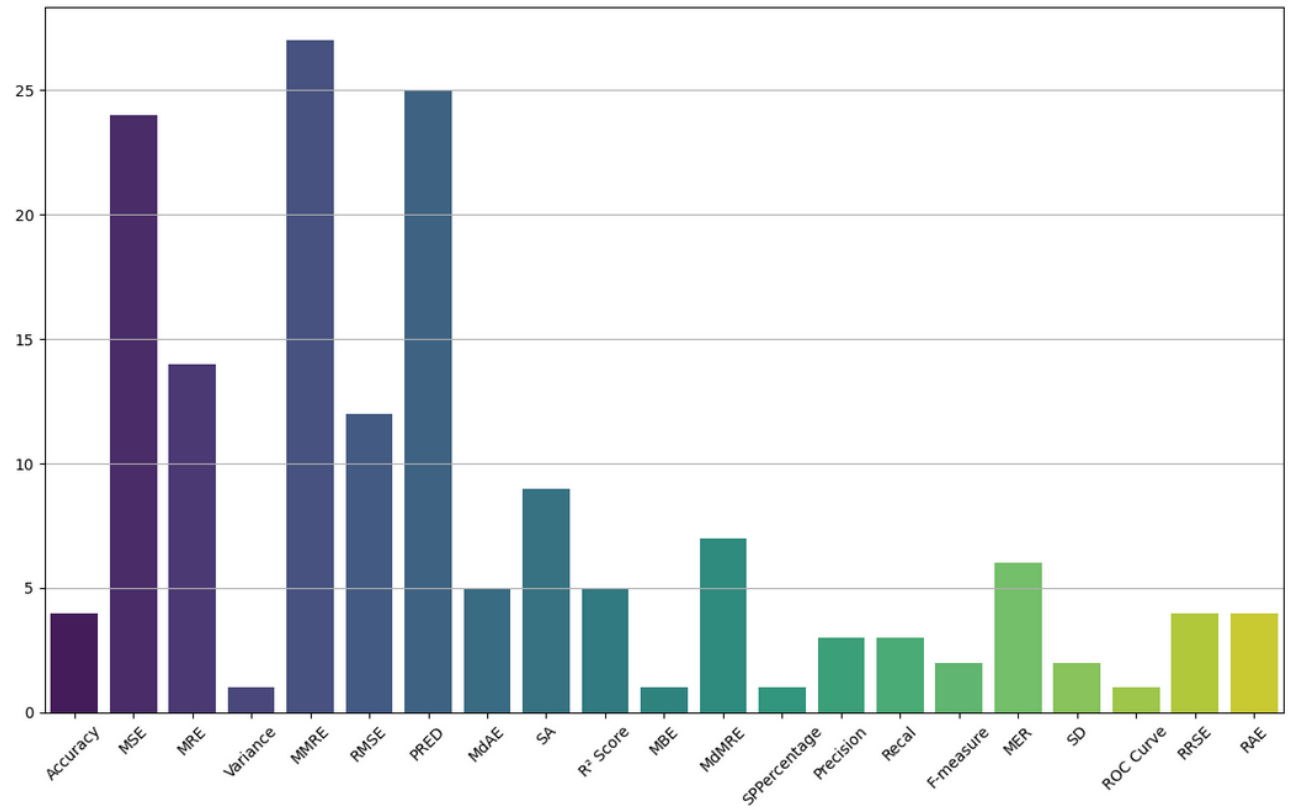

Figure 9. Assessment Metrics.

to generalize to commercial environments, restricting their practical utility. This highlights the pressing need for broader dataset diversity, which remains an open challenge for future research.

RQ3: What machine learning algorithms were used in these studies? Artificial Neural Networks (ANNs), including variants such as Recurrent Neural Networks (RNNs), appear most frequently in the reviewed literature, followed by models like Random Forest and Support Vector Machines (SVM). This trend suggests a preference for algorithms capable of learning complex patterns from textual data. Although the mapping primarily focused on algorithms, a qualitative analysis of the reviewed studies indicates limited integration of advanced text representation techniques, such as NLP-based embeddings. While this was not quantitatively measured, the observation highlights a potential gap in the literature, which can be explored in future work.

RQ4: What evaluation metrics were used in these studies? The analysis reveals a wide range of evaluation metrics across studies, with the most prevalent being Mean Absolute Error (MAE), Mean Relative Error (MRE), and Mean Magnitude of Relative Error (MMRE). This diversity highlights a lack of standardization in assessment methodologies, which complicates benchmarking and prevents cumulative knowledge building across studies. Additionally, metrics such as RMSE, R^2, and others—including Precision, Recall, and F-measure—are often used in combination to evaluate model performance, reflecting the complexity of measuring accuracy in this domain. While regression-based metrics are predominant, classification-oriented metrics remain underexplored, suggesting an opportunity for alternative modeling strategies. The current work follows the regression perspective, in alignment with the dominant approaches in the literature.

Key Findings from the Systematic Mapping

- **Growing interest in the field:** there has been a steady increase in the number of publications on effort estimation using machine learning, highlighting that the topic has been attracting increasing attention over the years.
- **Geographical concentration:** most studies were conducted in North America, Europe, and Asia, revealing significant gaps in regions such as Latin America and Africa, which limits the diversity and generalizability of the findings.
- **Dependence on open-source datasets:** a large portion of the works relied on open-source datasets, which favors the reproducibility of experiments. However, the lack of data from commercial projects may compromise external validity in industrial environments. Even when commercial datasets are available, they are often encrypted, making access and reproducibility difficult.
- **Predominance of neural networks:** ANNs and their variations (such as RNNs) were the most frequently used algorithms, followed by Random Forest and SVM. Nonetheless, qualitatively, there is a lack of adoption of more sophisticated language models.
- **Heterogeneous metrics:** a wide diversity of evaluation metrics (MAE, MRE, MMRE, RMSE, R^2, among others) was observed, without standardization of evaluation criteria, which hinders consistent comparisons across studies.
- **Research gaps:** the literature still lacks (i) more studies in industrial contexts, (ii) greater geographical diversity, (iii) standardization of evaluation metrics, and (iv) exploration of modern text representation techniques — the latter being precisely the central contribution of the present work.

4.7 Threats to validity

To conduct this systematic mapping study, was followed the protocol proposed by [Petersen *et al*., 2015], which ensures a research process aligned with principles of generality and descriptive validity. Despite this structured approach, some threats to validity must be acknowledged, particularly those related to the search terms and the inclusion and exclusion criteria applied.

A single search string was used across all selected databases. However, due to the unique characteristics and search mechanisms of each database, variations in the results may have occurred if customized search strings had been employed for each platform. This uniformity, while ensuring consistency, may have limited the retrieval of potentially relevant studies.

Another significant limitation involves the availability of time and resources. The mapping was conducted by a single researcher and reviewed solely by their supervisor. This limited involvement may have introduced bias in the study selection process and in the quality assessment of the included articles. The absence of a broader review team reduces the likelihood of detecting selection errors or inconsistencies, thereby impacting the overall reliability of the findings.

4.8 Conclusion of Systematic Mapping

The objective of this systematic mapping study was to analyze research that applies machine learning techniques to software effort estimation based on requirements. The analysis focused on the number of published studies, the machine learning approaches adopted, the primary datasets used for training and testing, and the evaluation metrics employed to assess model performance. To conduct this investigation, was used a hybrid search strategy, consulting the SCOPUS, IEEE, and ACM Digital Library databases. Additionally, was applied the backward snowballing technique to the references of each selected study, resulting in a total of 73 publications analyzed. The main contributions of this mapping include: (i) the synthesis of existing knowledge on the use of machine learning for effort estimation, covering key algorithms, datasets, and evaluation metrics while also highlighting the growing volume of research in recent years and (ii) the identification of research gaps that point to opportunities for future exploration and advancement in the field. Moreover, this study offers a historical perspective on how the state of the art has evolved with the introduction and refinement of new techniques and methodologies. One of the central challenges in conducting this mapping lies in the careful definition of selection criteria. Distinguishing between studies with similar scopes yet based on requirements with distinct characteristics demands

particular attention. A technique that performs well for a specific type of requirement may not yield satisfactory results for another, underscoring the need for contextual sensitivity in model evaluation. As a natural continuation of this work, it is suggested to expand the study to include newly published research and emerging techniques. This would allow for an updated and comprehensive overview of the current state of the art in software effort estimation, with a focus on machine learning approaches applied to software requirements.

5 User story estimations using NLP and DL

This section describes the model development process using the FastText and XLNET algorithm for feature extraction and model deep learning for inferring user story estimates. The process is divided into five stages: (1) data collection, (2) data pre-processing, (3) feature extraction, (4) modeling and inference, and (5) model evaluation. **Figure 10** presents a summary of the role of the model proposed here. Subsequently, each stage of the process is briefly presented.

5.1 Data Colletion and Preprocessing

The dataset used in this study was originally proposed by [Choetkiertikul *et al*., 2018] and consists of 23,313 user stories collected from 16 open-source projects. These textual requirements, generally managed through Jira, include a title, description, and effort estimation, the latter expressed as story points. **Figure 11**illustrates an example of the dataset used.

The effort estimates do not follow a fixed scale, such as the Fibonacci sequence commonly used in Planning Poker (e.g., 1, 2, 3, 5, 8, 13, 21, 40, 100). Instead, they reflect the actual time and work required to complete a requirement, measured from the "in progress" to the "resolved" state. Thus, this study approaches the estimation as a regression problem rather than a classification.

Two corpus were used:

- Corp_SE: composed of the labeled dataset with 23,313 user stories, used for training and evaluating the inference model.
- CorpPret_SE: consists of 302,962 unlabeled textual software requirements, used for unsupervised fine-tuning of the language model.

Table 8 provides an overview of the corpus.

Table 8. Corpora used in the experiments [Fávero *et al*., 2022]

Corpus	Specification	Application	Labeled
Corp_SE	23,313 user stories from 16 large open-source projects (e.g., Apache, Atlassian, Moodle, Spring).	Training and testing	Yes
Corp-Pret_SE	302,962 software requirements from various open-source projects (e.g., Apache, Moodle, Mesos).	Fine-tuning	No

The data was preprocessed to ensure textual consistency and reduce noise. The following steps were performed:

- Removal of HTML tags, special characters, stopwords, and numbers;
- Conversion to lowercase;
- Tokenization of user stories.

This preprocessing was applied to both corpus. Additionally, to align this study with related works such as [Fávero *et al*., 2022] and [Kassem *et al*., 2023], no normalization of story point values was applied.

Figure 12 shows the distribution of story points in Corp_SE, which is concentrated at lower values (1 to 8), with higher points occurring less frequently.

5.2 Features Extraction

This step aims to apply language models to textual requirements, aiming to generate a vector representation of each requirement in the form of embeddings.

To carry out this activity, the FastText and XLNet models will be used in two versions: with the suffix $base$, which corresponds to a model pre-trained with generic data, and with the suffix SE, which will go through pre-training and tuning using a corpus specific to the software requirements context, called corpPret_SE, as shown in **Table 8**.

With this, both models will be used to extract textual features, allowing the generation of the vector representation of each word as embeddings. As a result, each textual requirement in the corpus will be represented in matrix format, with dimensions adjusted according to the language model used.

The FastText model will be applied in its 300-dimensional version, while XLNet will use, by default, 768 dimensions. The textual representation in the format of embeddings will be used as input to the inference model during the training and testing stages and, subsequently, in inference.

5.3 Textual representation model

The process of extracting textual features is one of the most important steps before sending the data to the model, as there is a need to represent textual data in embedding formats so that the model can recognize the real meaning of each textual requirement. In order to compare context-free and contextualized models, the FastText and XLNET models were selected, which were explained in the Background session. FasText is a context-free model, that is, it represents words statically, ignoring the context in which they appear, unlike the XLNET model, which uses a contextualized approach. However, FastText is known for being a computationally efficient model both during training and testing, in addition to being simple to use and implement. In addition, it is a model that works with skip-grams, that is, while the standard model focuses on representing words as static vectors, skip-grams capture neighboring words from a central word [Bojanowski *et al*., 2017]. In this work, we used the FastText model with the skip-gram technique, previously trained on Wikipedia data in more than 157 languages, provided by the Hugging Face platform.

On the other hand, the contextualized XLNet model, also available on Hugging Face, assigns meaning to words considering the general context of the sentence, using a permuted

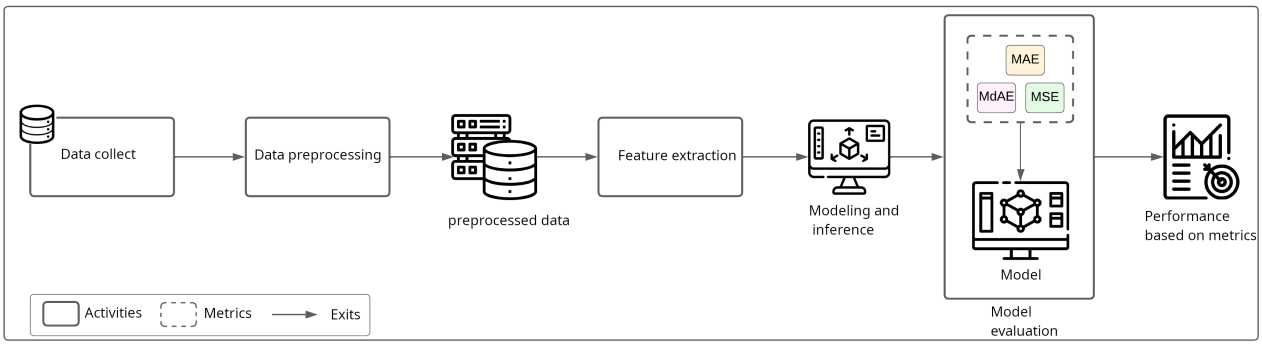

Figure 10. Pipeline of inference model

	issuekey	title	description	storypoint
0	TISTUD-6	Add CA against object literals in function inv...	{html} <p>The idea here is that if our met...	1
1	TISTUD-9	Update branding for Appcelerator plugin to App...	{html} <p>At least fix feature icons, asso...	1
2	TISTUD-11	Create new JSON schema for SDK team	{html} <p>Create JSON schema containing pr...	1
3	TISTUD-13	Create Project References Property Page	{html} <p>Create property page for project...	1
4	TISTUD-16	New Desktop Project Wizard	{html} <p>Desktop (need to convert existin...	1
...	...	...	...	...
97	TISTUD-348	Add pings to HTML5 events	Send analytic pings for project creation, laun...	1
98	TISTUD-355	Loading bar displaces Login / Cancel buttons a...	After clicking Login, the Loading bar bumps bo...	5
99	TISTUD-357	Splash screen jumps	When you first boot Titanium Studio, there is ...	5

Figure 11. Example of data from the used dataset

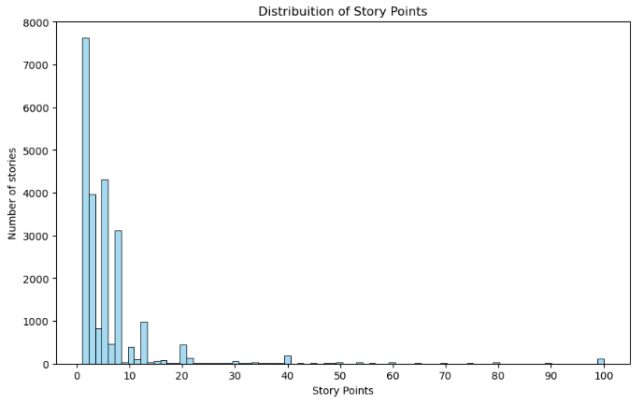

Figure 12. Distribution of story points in Corp_SE

autoregressive learning mechanism. This makes XLNet more robust compared to traditional models, although also more demanding in terms of computational cost. [Yang, 2019].

5.4 Fine-tuning process

The fine-tuning process consists of subjecting the language model to unsupervised training on a specific corpus, in the case of software requirements, so that the model learns the words used in the context. To carry out this process, the corpusPret_SE database was used, as described in **Table 8**. **Table 9** presents the configurations used for each type of model. The training and testing of both models were selected in a tested way, without using optimization methods.

5.5 Modeling and Inference

After pre-processing and feature extraction, this stage incorporates training and testing data into the model. For this, a DL architecture was used (**Figure 13** and **Figure 14**) that makes point predictions per story, following the same architecture proposed by Fávero et al. [Fávero *et al.*, 2022].

Table 9. Specification of fine-tuning process

Model	FastText	XLNET
Base Model	Pre-trained on Wikipedia data in over 157 languages	Pre-trained on large corpora such as BookCorpus and Wikipedia
Fine-tuning Corpus	CorpPret_SE	CorpPret_SE
Fine Tuning Technique	Unsupervised	Unsupervised
Model parameters	lr=1e-5 epochs=1 wordNgrams=2 dim=300	lr = 1e-5 batch_size=16 max_len=128 epochs=1
Library used	fasttext	transformers torch
Generated Model	fasttext_SE	XLNET_SE

Layer (type)	Output Shape	Param #
embedding_21 (Embedding)	(None, 768)	17,904,384
lstm_20 (LSTM)	?	0 (unbuilt)
dense_60 (Dense)	?	0 (unbuilt)
dense_61 (Dense)	?	0 (unbuilt)
dense_62 (Dense)	?	0 (unbuilt)

Figure 13. DL architecture with pre-trained embedding layer using XLNET

The model's input layer receives a vector of words from extracting textual features. We use the LSTM layer to represent texts in sequence. Afterward, the sentences will be subjected to two dense layers with non-linear activation functions, containing between 50 and 100 hidden layers, ending with a linear regression layer [Fávero *et al.*, 2022]. After the story point prediction has been carried out, the result will be evaluated based on the evaluation metrics presented in the next section.

The cross-validation technique was used to conduct this experiment, which aims to divide the database into *k* distinct parts to carry out tests and validations. In this execution, the value *k*=10 was adopted, where the training and testing process is repeated ten times. In each iteration, nine parts are used for training, while one part is reserved for testing.

5.6 Model Evaluation

To evaluate software effort estimation models based on regression models, some studies, such as [Choetkiertikul *et al.*, 2018], [Fávero *et al.*, 2022], [Kassem *et al.*, 2023], suggest the use of the metrics MAE, MdAE, and MSE. Equation 1 represents the MAE metric, where N represents the textual requirement, in this case, the user story. *Actual_eff* represents the current effort already represented, and *estimated_eff*

Layer (type)	Output Shape	Param #
embedding (Embedding)	(None, 300)	6,993,900
lstm (LSTM)	?	0 (unbuilt)
dense (Dense)	?	0 (unbuilt)
dense_1 (Dense)	?	0 (unbuilt)
dense_2 (Dense)	?	0 (unbuilt)

Figure 14. DL architecture with pre-trained embedding layer using FastText

represents the estimate made by the model.

$$MAE = \frac{1}{N}\sum_{i=1}^{N}|actual_eff - estimated_effi| \quad (1)$$

Equation 2 presents the MdAE metric, similar to the previous metric, but calculating the median. Finally, Equation 3 presents the MSE metric.

$$MdAE = median|actual_eff - estimated_effi| \quad (2)$$

$$MSE = \frac{1}{N}\sum_{i=1}^{N}(actual\_eff - estimated\_eff_i)^2 \quad (3)$$

Briefly, the MAE metric calculates the average error between the model estimate and the correct value labeled in the testbed. The MdAE metric follows the same principle but aims to calculate the median error between the estimate and the expected value. Lastly, the MSE metric calculates the mean squared error between the predicted value and the expected value[Fávero *et al*., 2022].

6 Experimental Study

This section details the experiment conducted to evaluate the accuracy of the FastText and XLNet models for estimating user stories from a real database. In this study, the guidelines recommended by [Wohlin *et al*., 2012] were used.

6.1 Objective

The Goal-Question-Metric (GQM) model [Basili and Weiss, 1986] was used to define the objectives of the experiment, summarized as follows: *"Analyze the FastText and XLNet pre-trained models for the purpose of evaluating their effectiveness in software effort estimation, from the perspective of the experimenter, in the context of a large real-world user story dataset."*

6.2 Research Questions

To achieve the goal, we seek to investigate the following RQs:

$RQ1$**: Would embedding models generated by context-less methods (i.e. FastText) be effective as models generated by contextual- ized methods (i.e. XLNET)?**

RQ_2**: Can a pre-trained embedding model, in a generic way, achieve the same efficiency as a model pre-trained using a specific corpus?**

$RQ3$**: In terms of computational cost, which of the two models was faster and more computational efficient during the fine-tuning process?**

$RQ4$**: The developed inference model achieves better results than related work?**

The evaluation metrics are directly aligned with each research question: MAE, MdAE, and MSE address RQ1 and RQ4 by measuring prediction accuracy, differences in MAE and MdAE before and after domain-specific fine-tuning address RQ2, and training time and memory consumption address RQ3. This mapping ensures full coherence with the GQM model.

6.3 Variables

- **Independent Variable:** Type of pre-trained model (FastText, XLNet).
- **Dependent Variables:** Prediction accuracy (measured by MAE, MdAE, MSE), computational efficiency (training time, memory usage).

6.4 Evaluation Metrics

The metrics used to evaluate the prediction accuracy of the models were: **Mean Absolute Error (MAE)**, **Median Absolute Error (MdAE)** and **Mean Squared Error (MSE)**. These metrics were selected due to their effectiveness in evaluating continuous prediction results, in addition to being the most used in related works, in addition to using computational cost and training time metrics to analyze the model's performance.

The choice of FastText and XLNet is grounded in their complementary nature: FastText represents non-contextual embeddings widely used in effort estimation studies, while XLNet provides contextualized representations capable of capturing long-range semantic dependencies. Their inclusion allows a direct comparison between contextualized and non-contextualized embeddings within the same architecture. The metrics MAE, MdAE, and MSE were selected because they are the most widely adopted in the effort estimation literature and allow comparison with related work, while also providing complementary perspectives on absolute and squared error behavior.

6.5 Experimental Procedure

To address the RQs, the experiment was conducted in the following stages:

1. **Dataset Selection:** The dataset used in this study was provided by Fávero et al. [Fávero *et al*., 2022], comprising 23,313 real-world user stories.
2. **Preprocessing:** The user stories underwent a series of preprocessing steps, including:
   - Removal of HTML tags, special characters, and stopwords;
   - Conversion to lowercase;
   - Tokenization of textual content.
3. **Model Configuration:** To ensure a fair comparison between embeddings, both FastText and XLNet were

integrated into the same neural architecture based on Long Short-Term Memory (LSTM) networks.

4. **Training:** The model training followed the same configuration as described in [Fávero *et al*., 2022], ensuring that results could be compared under equivalent experimental conditions.
    - A 10-fold cross-validation procedure was adopted to ensure robust evaluation;
    - The embedding layer was initialized with pre-trained vectors specific to each experiment (FastText or XLNet);
    - Models were trained for 30 epochs with a batch size of 128;
    - The Adam optimizer was used with a Mean Squared Error (MSE) loss function, and Mean Absolute Error (MAE) was the primary evaluation metric;
    - An early stopping mechanism with a patience of 10 epochs was employed to prevent overfitting, restoring the best weights obtained.

    The main hyperparameters used in the experiments were as follows: FastText embeddings were trained with 300 dimensions, window size 5, and negative sampling of 10. XLNet (base) was fine-tuned with a learning rate of 2e−5, batch size 16, and 3 epochs. The LSTM architecture employed 256 units, dropout of 0.3, ReLU activation, and the Adam optimizer with a learning rate of 1e−3. These settings follow established practices in the literature and were defined based on preliminary tuning to ensure stable training.
5. **Evaluation:** Model performance was assessed through multiple regression metrics, including MAE, Median Absolute Error (MdAE), and MSE. Computational resource consumption was also recorded.
6. **Comparison:** The results were compared to evaluate the effectiveness and efficiency of each embedding strategy in terms of prediction accuracy and resource usage.

6.6 Execution Environment

Experiments were run on a machine with an Intel Core i7-11800H CPU, NVIDIA RTX 3070 GPU, 32GB RAM, using Ubuntu 24.04.

7 Results

In this section, the results obtained when carrying out the experiment with each type of algorithm will be presented, in addition to also comparing them with related works, such as [Fávero *et al*., 2022] and [Kassem *et al*., 2023].

RQ_1**: Would embedding models generated by context-free methods (e.g., FastText) be as effective as embedding models generated by contextualized methods (e.g., XLNET)?**

To address this question, we analyzed the results obtained by integrating embeddings generated by both context-free and contextualized methods into the same deep learning architecture. Specifically, the FastText and XLNET models were compared under equivalent conditions, considering their generic versions (FastText_Base and XLNET_Base).

Table 10. Evaluation of the inference model using FastText and XLNet (without fine-tuning))

Abordagem	Modelo	MAE	MSE	MdAE	Tempo
Context-less	FastText_Base	4.04 ± 2.15	97.05 ± 127.09	2.19	94 min
Context	XLNET_Base	3.77 ± 0.15	79.94 ± 11.88	1.93	94 min

The results, presented in **Table 10**, indicate that the contextualized model (XLNET_Base) consistently outperformed the context-free model (FastText_Base) across all evaluation metrics. Specifically, it achieved approximately 6.7% better performance in terms of Mean Absolute Error (MAE), a 17.6% improvement in Mean Squared Error (MSE), and an 11.9% reduction in Median Absolute Error (MdAE).

These findings are consistent with previous studies highlighting the effectiveness of contextual embeddings in tasks that require deeper semantic understanding [Devlin *et al*., 2018; Yang, 2019]. However, they also reinforce the notion that lightweight models, when properly tuned, can achieve satisfactory performance in more narrowly scoped applications, such as story point estimation.

In summary, models such as XLNET offer greater capability to capture the complexity of natural language found in user stories. However, lighter approaches based on models such as FastText can be considered, when trained on a specific corpus. To address this issue, RQ2 aims to investigate the behavior of each model when subjected to a fine-tuning process.

$RQ2$**: Would embedding fine-tuned models in user stories improve effort estimation results compared to their generic versions?**

To investigate the impact of fine-tuning on embedding models in the context of story point estimation, we compared the performance of each model in its generic version and in its fine-tuned version trained on user stories. The goal was to determine whether adapting pre-trained embeddings to the software requirements domain would increase their effectiveness when integrated into a deep learning architecture.

Table 11. Evaluation of the inference model using FasText and XLNET

Approach	Model	MAE	MSE	MdAE	Time
Context-less	FastText_Base	4.04 ± 2.15	97.05 ± 127.09	2.19	94 min
	FastText_SE	3.84 ± 0.16	81.86 ± 8.81	1.86	36 min
Context.	XLNET_Base	3.77 ± 0.15	79.94 ± 11.88	1.93	94 min
	XLNET_SE	3.94 ± 0.15	81.33 ± 10.53	1.94	95 min

The results, summarized in **Table 11**, show that fine-tuning had a positive effect on the context-free model (FastText). After fine-tuning on user stories (FastText_SE), the model showed performance gains across all metrics, including a 6.9% reduction in MAE, 13.4% in MSE, and 15.6% in MdAE. These improvements suggest that adapting word vectors to the vocabulary and stylistic patterns of user stories allows for better semantic representation, which positively impacts the model's ability to predict story points.

In contrast, the contextualized model (XLNET) showed minimal or no improvements after fine-tuning. The fine-tuned version (XLNET_SE) performed slightly worse than its generic counterpart (XLNET_Base) in terms of MAE and

MSE, with equivalent performance in MdAE. This behavior may indicate that contextualized models, due to their richer semantic structure and exposure to a large-scale corpus, are already well-suited for general tasks, including story point estimation. As a result, fine-tuning may offer limited benefits and even introduce slight overfitting when applied to a relatively small domain-specific dataset. From these observations, it follows that fine-tuning is particularly effective for context-free models such as FastText, while for powerful contextual models such as XLNET, the gains are marginal and may not justify the additional computational cost in this context. This distinction is important for practitioners choosing between pre-trained embeddings and those tailored to specific software engineering domains.

*RQ*3**: In terms of computationally cost, which of the two models was faster and more computational efficient during the fine-tuning process?**

To assess the computational efficiency of the models during fine-tuning, metrics such as CPU and GPU usage, memory consumption, and total training time were recorded, as summarized in **Table 12**. The results indicate that FastText had a substantially lower computational cost, completing the fine-tuning process in 48 minutes, while XLNET required 132 minutes, almost three times longer.

Table 12. Computational resources during the fine-tuning process

Resource	FastText(CPU)	XLNET(GPU)
Time	48m18s	132m0s
Initial CPU Usage	7.1%	8.8%
CPU Usage After	92.9%	7.4%
Initial Used Memory	13.39 GB	3.89 GB
Memory Used After	18.17 GB	3.90 GB
Initial GPU Usage	0.0%	12.0 MB
GPU Usage After	0.0%	4187.0 MB

FastText operated entirely on the CPU, utilizing a larger share of RAM (up to 18.17 GB), but without GPU consumption. This is in line with the model design, which favors lightweight execution and efficiency on limited hardware, making it suitable for infrastructure-constrained scenarios (e.g., on-premises servers, edge devices). On the other hand, XLNET has offloaded most of the computation to the GPU, consuming over 4 GB of GPU memory, which, while alleviating CPU and RAM usage, requires access to specialized hardware and increases energy and financial costs.

While XLNET has demonstrated improved predictive performance (as discussed in RQ1 and RQ2), this improvement comes at the expense of significantly higher computational overhead. Therefore, in real-world applications, especially in industrial environments where models need to be trained or retrained frequently, or deployed at scale, the trade-off between performance and cost becomes critical.

In this context, FastText may represent a more cost-effective option, especially when high-frequency updates, rapid iteration, or deployment in low-power environments are required. Even with slightly lower predictive performance, the gain in speed and accessibility may outweigh the benefits of a marginal increase in accuracy provided by more computationally intensive models such as XLNET.

Therefore, the results of RQ3 not only reveal which model is computationally cheaper, but reinforce the importance of aligning model selection with practical constraints such as infrastructure availability, energy efficiency, and deployment environment.

*RQ*4**: Does the developed inference model achieve better results than related work?**

To answer this question, we evaluated the proposed inference model against approaches from related works. First, we compared the model, based on the XLNET language model, with the architecture proposed by [Fávero *et al*., 2022], which used BERT embeddings in a similar deep learning setting. As shown in **Table 13**, the proposed model achieved superior performance, reducing MAE by 11%, MSE by 7%, and MdAE by 16%. These improvements suggest that, among contextualized language models, XLNET-based embeddings provide better representations for the user story estimation task.

Table 13. Evaluation of the inference model using XLNET and BERT

Approach	Model	MAE	MSE	MdAE
Context	XLNET_Base	3.77 ± 0.15	79.94 ± 11.88	1.93
Context	BERT_SE	4.25 ± 0.17	86.15 ± 1.66	2.3

The proposed model was also compared with the approach proposed by [Kassem *et al*., 2023], which used GloVe embeddings (a non-contextual model) and a HAN (Hierarchical Attention Network) architecture. As shown in **Table 14**, the HAN model obtained a substantially lower MAE (1.86) and MdAE (0.69) result than the one proposed in this research, outperforming the results in these metrics.

Table 14. Evaluation of the inference model using XLNET and HAN

Approach	Model	MAE	MdAE
Context	XLNET_Base	3.77 ± 0.15	1.93
Context-less	HAN(GloVe)	1.86	0.69

These results indicate that, although the proposed model outperforms previous contextualized approaches, it still does not outperform the context-free HAN(GloVe) model. This result can be attributed to several factors: (i) the HAN architecture is particularly effective in capturing hierarchical relationships in text, which can be crucial in modeling user stories; (ii) GloVe embeddings, although non-contextual, can better align with the hierarchical encoding strategy; and (iii) the architecture used, although based on deep learning and contextual embeddings, may require additional adjustments or structural improvements to better leverage the full potential of contextualized representations.

Despite these limitations, the proposed model remains competitive because it integrates contextual embeddings capable of generalizing different text scenarios and can benefit from additional improvements. Furthermore, the use of contextual embeddings allows for better interpretability and adaptability in downstream tasks, which is valuable in real agile estimation contexts. Future work could investigate hybrid approaches that combine the strengths of hierarchical architectures with contextual embeddings to further improve performance.

8 ESApp Web Application

To improve the applicability of the tool developed in this project, a web interface called ESApp was implemented. The goal is to assist software developers during the effort estimation phase of user stories, providing a second opinion, even if the model does not achieve 100% accuracy in its predictions.

To make this possible, the predictive model was serialized using the `joblib` library, and the user interface was developed with the `Streamlit` framework. **Figure 15** presents the architecture of the web application, while **Figure 16** illustrates the tool's home screen.

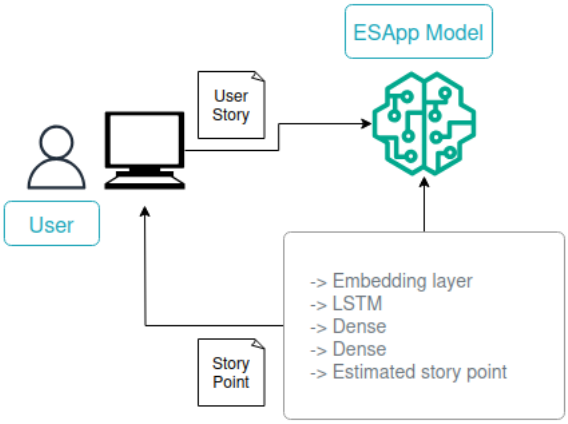

Figure 15. Architecture of web application

The architecture illustrates how the application performs effort estimation. Initially, the user inputs a user story description through the web interface and submits it for estimation. The input is then processed by the proposed model: the text is first embeddable through the embedding layer, followed by sequential processing using an LSTM layer. Subsequently, the output passes through two dense layers to generate the final effort estimate, which is then returned to the user.

The application supports both individual and batch estimations. **Figure 16** demonstrates the individual estimation mode, where a single user story is input and the model returns the estimated story points. **Figure 17** shows batch mode, where multiple user stories are submitted via a CSV file, and the application returns estimated values for each entry.

The application is publicly accessible and can be tested via the following link: ESApp Web Application.

8.1 Tool Validation with Software Developers

This section presents a validation study of the ESApp tool conducted with software developers. Unlike the quantitative experiments described in the previous sections, this evaluation focuses on the practical applicability, usability, and perceived usefulness of the tool in real-world scenarios.

Participants and Methodology Seven software developers from different software houses participated in the study. The selection criteria aimed to include professionals with diverse levels of experience to gather a broad perspective on the tool's applicability. Among the participants, 28.6% identified as junior, 42.9% as mid-level, and 28.6% as senior developers. All participants currently work in software development.

Before starting, participants were informed about the study objectives and gave their consent to participate by answering the questionnaire, ensuring ethical considerations were met.

Procedure Each developer was asked to use the ESApp tool to estimate at least three user stories. After completing the estimation tasks, participants answered a structured questionnaire designed to capture their experience and opinions regarding the tool. The questionnaire included the following points:

1. Confirmation of their work in software development.
2. Their level of experience.
3. Previous use of effort estimation tools for textual requirements.
4. Perceived accuracy of the estimates produced by ESApp (rated on a 1 to 5 scale).
5. Usefulness of the tool in their daily work.
6. Positive aspects of the tool.
7. Suggestions for improvements.

The estimation activity took approximately 10 minutes, followed immediately by questionnaire completion.

Results

- **Experience and Background:** All participants work in software development. The experience levels were distributed as described above, as presented in **Figure 18**.
- **Previous Tool Usage:** 71.4% of the participants had never used effort estimation tools for textual requirements, indicating a mostly fresh user base, as presented in **Figure 19**.
- **Tool Accuracy Rating:** On a scale from 1 (lowest) to 5 (highest), the average accuracy rating given by participants was 3.57, with most ratings between 3 and 5 (see **Figure 20**).
- **Perceived Usefulness:** All participants (100%) agreed that the tool could be useful in their daily work.
- **Positive Feedback:** Users highlighted the tool's simplicity, speed, agility, and ease of use as its main strengths.
- **Improvement Suggestions:** Participants suggested several enhancements, such as:
  - Providing explanations for the estimates generated to improve transparency.
  - Avoiding decimal values in story points, aligning with the common use of the Fibonacci sequence.
  - Adding support for the Portuguese language.
  - Making the underlying basis of the estimates more transparent.

These insights are valuable for guiding future iterations and improvements of the tool.

Discussion Although this validation study is limited by the small number of participants, it provides an initial positive indication of the acceptance and potential usefulness of the tool in software development environments. The feedback

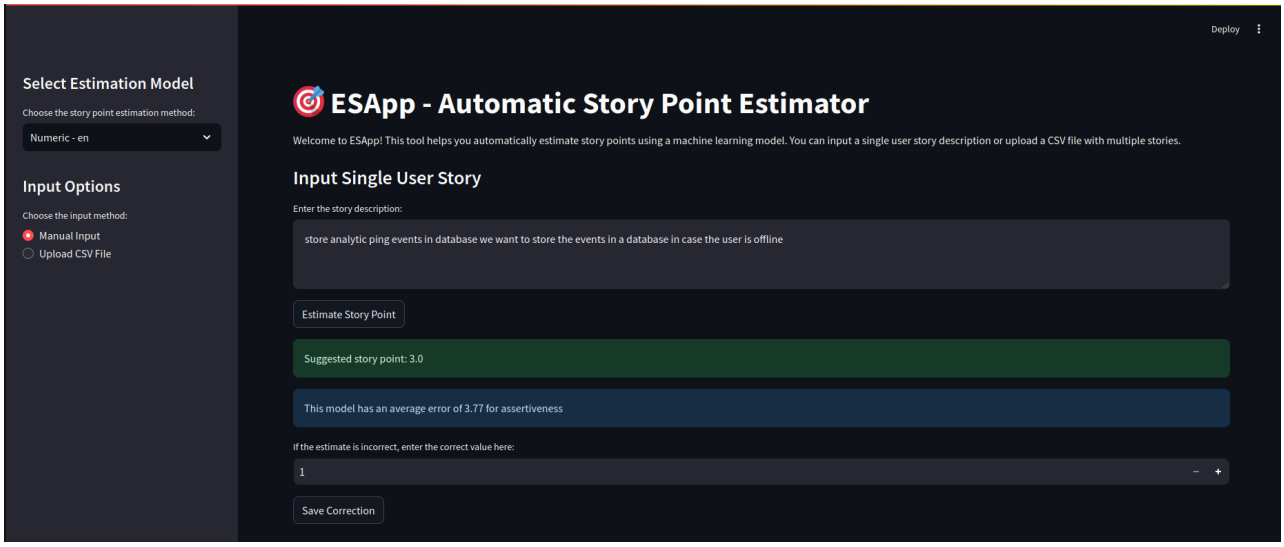

Figure 16. ESApp - Application User History Estimation.

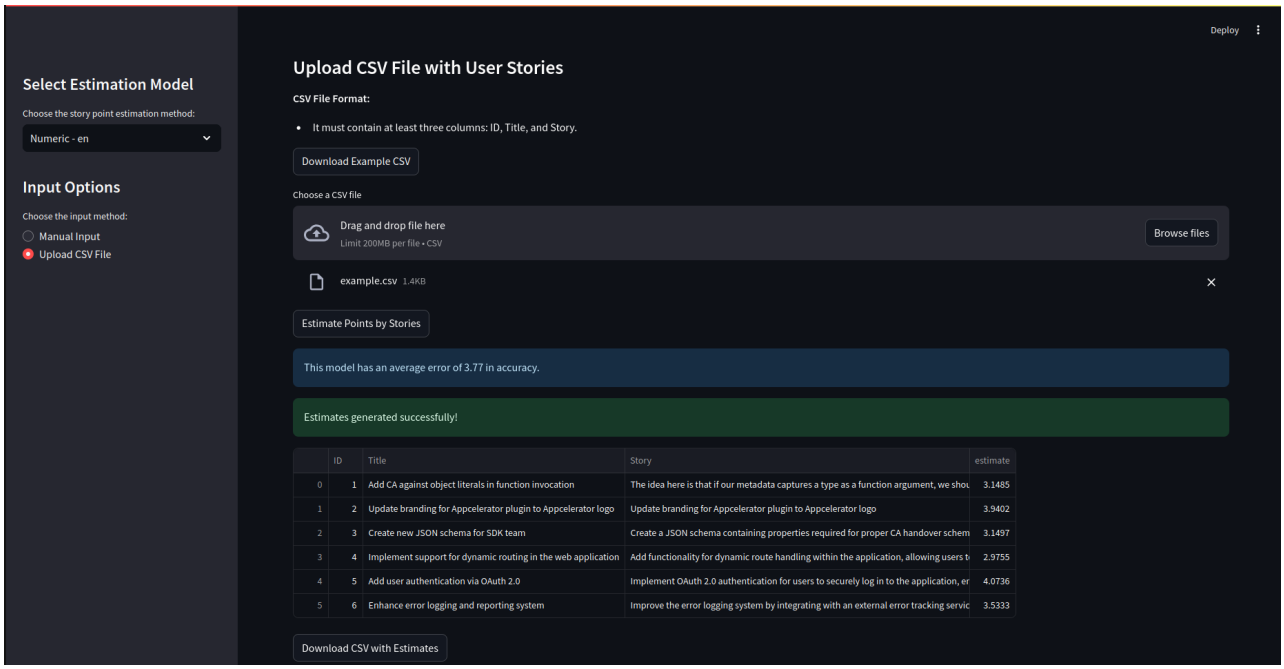

Figure 17. ESApp - estimating multiple user stories.

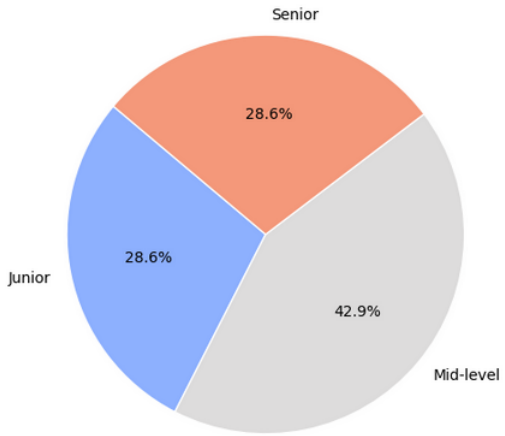

Figure 18. Experience level as a developer

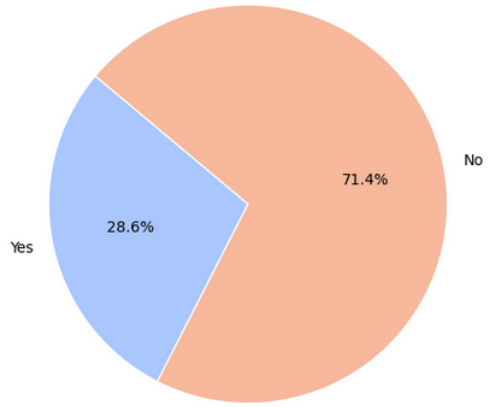

Figure 19. Use of software effort estimation tools.

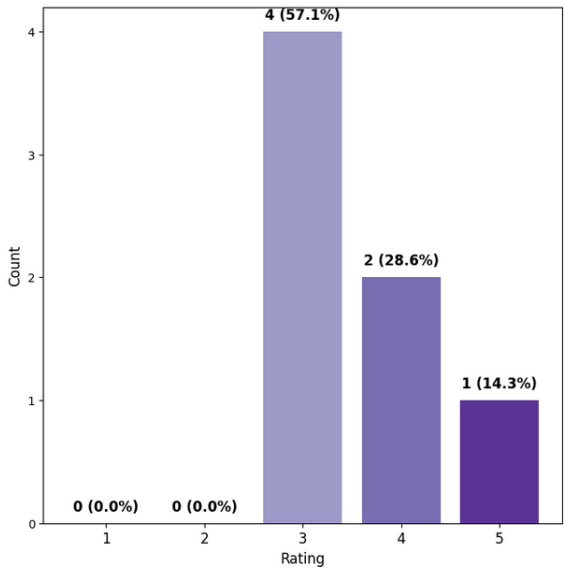

Figure 20. Tool assertiveness

highlights both strengths and areas for improvement, especially regarding interpretability and adaptation to practical workflows.

Future studies could expand the participant pool, include qualitative interviews, and conduct more rigorous statistical analyses to deepen understanding of the tool's impact and user satisfaction.

8.2 Reproducibility

Reproducibility is a fundamental aspect in the development of research projects, as it allows other researchers to reproduce the study and use its results as a basis for future work. As mentioned in previous sections, all databases used in this project were obtained from open source projects, including those originated by the work of [Fávero *et al*., 2022] and [Choetkiertikul *et al*., 2018].

The pre-processing techniques were previously detailed and, for this experiment, the FastText and XLNet language models were used. These models were chosen due to their ability to generate robust semantic representations for Portuguese text.

The development of the deep learning model, as well as the training and testing methods, were implemented using the Python language and the Visual Studio Code IDE. To ensure reproducibility, all necessary files and scripts are available in a public repository on GitHub, accessible via the following link: GitHub Repository.

8.3 Threats to Validity

Several threats to the validity of this study should be considered:

Construct Validity: The effort estimation model was trained using textual descriptions of user stories. Any inconsistencies, ambiguities, or lack of detail in these texts may affect the quality of the extracted features and, consequently, the prediction performance. Furthermore, the use of story points as ground truth, which is inherently subjective, may introduce variability into the model training process.

Internal Validity: Although care was taken in preprocessing the data and evaluating the model, biases may still exist in the way the data was labeled or divided into training and testing sets. For example, the distribution of story points is unbalanced and developers may not have followed a consistent estimation process, especially across different projects.

External Validity: The dataset used for training and validating the model was collected exclusively from 16 *open source* projects. While this contributes to reproducibility, it limits the generalizability of the results. Open source environments often differ from commercial projects in terms of scope, development processes, resource availability, and stakeholder demands. Therefore, further validation is needed in industrial settings to assess the effectiveness of the model in real commercial scenarios.

Conclusion Validity: Due to the limited number of participants (seven software developers) in the evaluation of the application, the feedback may not be representative of the developer community at large. Furthermore, the evaluation relied on subjective assessments of the accuracy and usefulness of the model, which, while valuable, are no substitute for large-scale empirical validation.

Despite these limitations, this study lays the foundation for the application of deep learning to effort estimation and demonstrates promising results that warrant future research and refinement of the approach.

9 Conclusion

This paper presented a comparative study between the XLNET and FastText language models, using the same deep learning architecture for the automatic estimation of user stories. The experiments demonstrated that, although both models presented similar performances in the MAE, MSE, and MdAE metrics, XLNET outperformed FastText, as evidenced by the results. Despite the minimal difference in the numbers, this similarity suggests that, for the specific task of estimating effort in user stories, XLNET's more robust contextual understanding can be a differentiator compared to other deep learning approaches, especially when applied to varied requirements and different projects. In this context, the XLNET model, even in its generic version, stood out as an important tool due to its contextualized nature and the robustness provided by the large volume of data on which it was trained. Furthermore, it is essential to consider that the choice between XLNET and FastText for software requirements effort estimation should take into account not only the accuracy of the estimates, but also practical factors, such as the available computational resources, the size and complexity of the user stories, and the language context of the application.

As perspectives for future work, it is suggested to explore the application of different machine learning algorithms and other inference models, including, for example, the HAN model [Kassem *et al*., 2023]. In addition, it would be valuable to compare the results of these models with estimates performed by experts in order to evaluate the accuracy and effectiveness of each approach. It is also recommended to expand the application of the developed model to commercial project databases, enabling a more in-depth analysis of its effectiveness and performance in different contexts and types of projects. This would allow a more comprehensive evaluation of the model's generalizability and its ability to adapt to real situations in software development.

This research contributes to software engineering by providing a comparative basis between two NLP techniques using deep learning, expanding the understanding of how they can be applied to improve effort estimates in software development. The use of the XLNET model proves to be a promising alternative when applied to software effort estimation based on textual requirements. However, this work highlights the importance of continuing research in this area, especially in exploring models in languages other than English, such as Brazilian Portuguese, which has a more complex grammatical structure. In addition, it highlights the need for practical validation of the model in industrial environments, ensuring that its predictions can be effectively applied to real projects.

10 Acknowledgements

The authors would like to thank the Graduate Program in Informatics (PPGI) – Cornélio Procópio, at the Federal University of Technology – Paraná (UTFPR), for supporting the development of this work as part of the Professional Master's Program.

11 Author's contribution

DOS was the main contributor to this study, having developed the study design, implemented the model, conducted the systematic mapping, analyzed the results, and written the manuscript. FCMS and ACCS contributed equally to the definition and planning of the study, as well as to the research, development, writing, and review processes. SRRS assisted in the writing and review stages. All authors read and approved the final version of the manuscript.

12 Competing interests

The authors declare that they have no competing interests.

13 Availability of Data and Materials

All data used in this study are publicly available at GitHub Repository.

References

Abadeer, M. and Sabetzadeh, M. (2021). Machine learning-based estimation of story points in agile development: Industrial experience and lessons learned. In *2021 IEEE 29th International Requirements Engineering Conference Workshops (REW)*, pages 106–115. IEEE. DOI: 10.1109/REW53955.2021.00022.

Abdukalykov, R. (2011). *A New Methodology for Quantifying the Impact of Non-Functional Requirements on Software Effort Estimation*. PhD thesis, Concordia University. Available at:`https://spectrum.library.concordia.ca/id/eprint/15142/`.

Abnane, I., Hosni, M., Idri, A., and Abran, A. (2019). Analogy software effort estimation using ensemble knn imputation. In *2019 45th Euromicro Conference on Software Engineering and Advanced Applications (SEAA)*, pages 228–235. IEEE. DOI: 10.1109/SEAA.2019.00044.

Abrahamsson, P., Fronza, I., Moser, R., Vlasenko, J., and Pedrycz, W. (2011). Predicting development effort from user stories. In *2011 International Symposium on Empirical Software Engineering and Measurement*, pages 400–403. IEEE. DOI: 10.1109/ESEM.2011.58.

AG, P. V., K, A. K., and Varadarajan, V. (2021). Estimating software development efforts using a random forest-based stacked ensemble approach. *Electronics*, 10(10):1195. DOI: 10.3390/electronics10101195.

Aranda, J. and Easterbrook, S. (2005). Anchoring and adjustment in software estimation. In *Proceedings of the 10th European software engineering conference held jointly with 13th ACM SIGSOFT international symposium on Foundations of software engineering*, pages 346–355. DOI: 10.1145/1095430.1081761.

Araujo, R. d. A., Oliveira, A. L., and Meira, S. (2017). A class of hybrid multilayer perceptrons for software development effort estimation problems. *Expert Systems with Applications*, 90:1–12. DOI: 10.1016/j.eswa.2017.07.050.

Arora, M., Sharma, A., Katoch, S., Malviya, M., and Chopra, S. (2021). A state of the art regressor model's comparison for effort estimation of agile software. In *2021 2nd International Conference on Intelligent Engineering and Management (ICIEM)*, pages 211–215. IEEE. DOI: 10.1109/ICIEM51511.2021.9445345.

Azzeh, M., Nassif, A. B., and Banitaan, S. (2018). Comparative analysis of soft computing techniques for predicting software effort based use case points. *Iet Software*, 12(1):19–29. DOI: 10.1049/iet-sen.2016.0322.

Azzeh, M., Nassif, A. B., and Minku, L. L. (2015). An empirical evaluation of ensemble adjustment methods for analogy-based effort estimation. *Journal of Systems and Software*, 103:36–52. DOI: 10.1016/j.jss.2015.01.028.

Baratto, G. J. (2022). Comparação entre os modelos pré-treinados gpt-3 e bert na estimativa de esforço de software por analogia a partir de requisitos textuais. B.S. thesis, Universidade Tecnológica Federal do Paraná. Available at:`https://repositorio.utfpr.edu.br/jspui/handle/1/30619`.

Basili, V. R. and Weiss, D. M. (1986). A methodology for collecting valid software engineering data. *IEEE Transactions on Software Engineering*, 10(6):728–738. DOI: 10.1109/TSE.1984.5010301.

Bengio, Y., Courville, A., and Vincent, P. (2013). Representation learning: A review and new perspectives. *IEEE transactions on pattern analysis and machine intelligence*, 35(8):1798–1828. DOI: 10.1109/tpami.2013.50.

Bhat, J. M., Gupta, M., and Murthy, S. N. (2006). Overcoming requirements engineering challenges: Lessons from offshore outsourcing. *IEEE software*, 23(5):38–44. DOI: 10.1109/ms.2006.137.

Bilgaiyan, S., Mishra, S., and Das, M. (2019). Effort estimation in agile software development using experimental validation of neural network models. *International Journal of Information Technology*, 11(3):569–573. DOI: 10.1007/s41870-018-0131-2.

Boehm, B., Abts, C., and Chulani, S. (2000). Software development cost estimation approaches—a survey. *Annals of software engineering*, 10(1-4):177–205. DOI: 10.1023/A:1018991717352.

Bojanowski, P., Grave, E., Joulin, A., and Mikolov, T. (2017). Enriching word vectors with subword information. *Transactions of the association for computational linguistics*, 5:135–146. DOI: 10.1023/A:1018991717352.

Braga, P. L., Oliveira, A. L., and Meira, S. R. (2007). Software effort estimation using machine learning techniques with robust confidence intervals. In *7th international conference on hybrid intelligent systems (HIS 2007)*, pages 352–357. IEEE. DOI: 10.1109/HIS.2007.56.

Britto, R., Freitas, V., Mendes, E., and Usman, M. (2014). Effort estimation in global software development: A systematic literature review. In *2014 IEEE 9th International Conference on Global Software Engineering*, pages 135–144. IEEE. DOI: 10.1109/ICGSE.2014.11.

Cambria, E. and White, B. (2014). Jumping nlp curves: A review of natural language processing research. *IEEE Computational intelligence magazine*, 9(2):48–57. DOI: 10.1109/MCI.2014.2307227.

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. DOI: 10.1109/TSE.2018.2792473.

Choi, S., Park, S., and Sugumaran, V. (2012). A rule-based approach for estimating software development cost using function point and goal and scenario based requirements. *Expert Systems with Applications*, 39(1):406–418. DOI: 10.1016/j.eswa.2011.07.029.

Cohn, M. (2005). *Agile estimating and planning*. Pearson Education. Book.

Costa, L. A. and Salvador, L. d. N. (2015). Ambiente de aprendizagem presencial e virtual integrados com a computação ubíqua: Um mapeamento sistemático da literatura. In *Memorias del XX Congresso Internacional de Informática Educativa, TISE*, volume 11, pages 211–220. Available at:`https://sol.sbc.org.br/index.php/sbie/article/download/26688/26507/`.

de Morais, R. A. (2021). Deep learning based models for software effort estimation using story points in agile environments. DOI: 10.7939/r3-jcf5-8x08.

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. (2018). Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*. DOI: 10.18653/v1/N19-1423.

Dhir, S., Kumar, D., and Singh, V. (2017). An estimation technique in agile archetype using story points and function point analysis. *International Journal of Process Management and Benchmarking*, 7(4):518–539. DOI: 10.1504/IJPMB.2017.086933.

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. DOI: 10.1016/j.jss.2017.01.027.

Fávero, E. M. D. B., Casanova, D., and Pimentel, A. R. (2022). Se3m: A model for software effort estimation using pre-trained embedding models. *Information and Software Technology*, 147:106886. DOI: 10.1016/j.infsof.2022.106886.

Fávero, E. M. D. B., Pereira, R., Pimentel, A. R., and Casanova, D. (2018). Analogy-based effort estimation: A systematic mapping of literature. *INFOCOMP Journal of Computer Science*, 17(2):07–22. Available at:`https://infocomp.dcc.ufla.br/index.php/infocomp/article/view/565`.

Fedotova, O., Teixeira, L., Alvelos, H., *et al*. (2013). Software effort estimation with multiple linear regression: Review and practical application. *J. Inf. Sci. Eng.*, 29(5):925–945. DOI: 10.6688/JISE.2013.29.5.8.

Fu, M. and Tantithamthavorn, C. (2022). Gpt2sp: A transformer-based agile story point estimation approach. *IEEE Transactions on Software Engineering*, 49(2):611–625. DOI: 10.1109/TSE.2022.3158252.

Ghosal, S. and Jain, A. (2023). Depression and suicide risk detection on social media using fasttext embedding and xgboost classifier. *Procedia Computer Science*, 218:1631–1639. DOI: 10.1016/j.procs.2023.01.141.

Goldberg, Y. (2022). *Neural network methods for natural language processing*. Springer Nature. DOI: 10.1007/978-3-031-02165-7.

Grenning, J. (2002). Planning poker or how to avoid analysis paralysis while release planning. *Hawthorn Woods: Renaissance Software Consulting*, 3:22–23. Available at:https://sewiki.iai.uni-bonn.de/_media/teaching/labs/xp/2005a/doc.planningpoker-v1.pdf.

Gultekin, M. and Kalipsiz, O. (2020). Story point-based effort estimation model with machine learning techniques. *International Journal of Software Engineering and Knowledge Engineering*, 30(01):43–66. DOI: 10.1142/S0218194020500035.

Gupta, N. and Mahapatra, R. P. (2022). Automated software effort estimation for agile development system by heuristically improved hybrid learning. *Concurrency and Computation: Practice and Experience*, 34(25):e7267. DOI: 10.1002/cpe.7267.

Haj-Yahia, Z., Sieg, A., and Deleris, L. A. (2019). Towards unsupervised text classification leveraging experts and word embeddings. In *Proceedings of the 57th annual meeting of the Association for Computational Linguistics*, pages 371–379. DOI: 10.18653/v1/p19-1036.

Hamouda, A. E. D. (2014). Using agile story points as an estimation technique in cmmi organizations. In *2014 agile conference*, pages 16–23. IEEE. DOI: 10.1109/AGILE.2014.11.

Hemrajani, M. V. N. and N, V. (2021). Predicting effort of agile software projects using linear regression, ridge regression and logistic regression. Available at:https://www.researchgate.net/publication/385600508_PREDICTING_EFFORT_OF_AGILE_SOFTWARE_PROJECTS_USING_LINEAR_REGRESSION_RIDGE_REGRESSION_AND_LOGISTIC_REGRESSION.

Hidmi, O. and Sakar, B. E. (2017). Software development effort estimation using ensemble machine learning. *Int. J. Comput. Commun. Instrum. Eng*, 4(1):143–147. DOI: 10.15242/IJCCIE.E0317026.

Hirschberg, J. and Manning, C. D. (2015). Advances in natural language processing. *Science*, 349(6245):261–266. DOI: 10.1126/science.aaa8685.

Hussain, I., Kosseim, L., and Ormandjieva, O. (2013). Approximation of cosmic functional size to support early effort estimation in agile. *Data & Knowledge Engineering*, 85:2–14. DOI: 10.1016/j.datak.2012.06.005.

Hussein, L. A., Nassar, K. A., and Naser, M. (2017). Recurrent neural network based prediction of software effort. *International Journal of Computer Applications*, 975:8887. DOI: 10.5120/ijca2017915664.

Jobera Editorial Team (2025). Software project failure statistics: Insights and trends. Available at:https://jobera.com/software-project-failure-statistics/. Accessed: 2025-04-19.

Jurafsky, D. and Martin, J. H. (2019). Speech and language processing (3rd (draft) ed.). Available at:https://web.stanford.edu/~jurafsky/slp3/.

Kanmani, S., Kathiravan, J., Kumar, S. S., and Shanmugam, M. (2007). Neural network based effort estimation using class points for oo systems. In *2007 International Conference on Computing: Theory and Applications (ICCTA'07)*, pages 261–266. IEEE. DOI: 10.1109/ICCTA.2007.89.

Kassem, H., Mahar, K., and Saad, A. A. (2023). Story point estimation using issue reports with deep attention neural network. *e-Informatica Software Engineering Journal*, 17(1). DOI: 10.37190/e-Inf230104.

Kaushik, A. and Singal, N. (2022). A hybrid model of wavelet neural network and metaheuristic algorithm for software development effort estimation. *International Journal of Information Technology*, 14(3):1689–1698. DOI: 10.1007/s41870-019-00339-1.

Kaushik, A., Tayal, D. K., and Yadav, K. (2020). A comparative analysis on effort estimation for agile and non-agile software projects using dbn-alo. *Arabian Journal for Science and Engineering*, 45(4):2605–2618. DOI: 10.1007/s13369-019-04250-6.

Keele, S. *et al*. (2007). Guidelines for performing systematic literature reviews in software engineering. DOI: 10.1109/LA-CCI.2017.8285683.

Khan, M. W. and Qureshi, I. (2014). Neural network based software effort estimation: a survey. *International Journal of Advanced Networking and Applications*, 5(4):1990. Available at:https://www.semanticscholar.org/paper/Neural-Network-based-Software-Effort-Estimation%3A-A-Khan-Qureshi/fe17a86aff21cb858f8f04837e0d097022ead2a2.

Khuat, T. T. and Le, M. H. (2016). An effort estimation approach for agile software development using fireworks algorithm optimized neural network. *Int. J. Comput. Sci. Inf. Secur.(IJCSIS)*, 14:122–130. Available at:https://www.researchgate.net/publication/306099986_An_Effort_Estimation_Approach_for_Agile_Software_Development_using_Fireworks_Algorithm_Optimized_Neural_Network.

Khuat, T. T. and Le, M. H. (2018). A novel hybrid abc-pso algorithm for effort estimation of software projects using agile methodologies. *Journal of Intelligent Systems*, 27(3):489–506. DOI: 10.1515/jisys-2016-0294.

Kitchenham, B. and Charters, S. (2007). Guidelines for performing systematic literature reviews in software engineering. Technical Report EBSE-2007-01, Keele University and Durham University. Available at:https://legacyfileshare.elsevier.com/promis_misc/525444systematicreviewsguide.pdf.

Kocaguneli, E., Menzies, T., and Keung, J. W. (2011). On the value of ensemble effort estimation. *IEEE Transactions on Software Engineering*, 38(6):1403–1416. DOI: 10.1109/TSE.2011.111.

Kultur, Y., Turhan, B., and Bener, A. B. (2008). Enna: software effort estimation using ensemble of neural networks with associative memory. In *Proceedings of the 16th ACM SIGSOFT International Symposium on Foundations of software engineering*, pages 330–338. DOI: 10.1145/1453101.1453148.

Kumar, P. S., Behera, H. S., Kumari, A., Nayak, J., and Naik, B. (2020). Advancement from neural networks to deep learning in software effort estimation: Perspective of two decades. *Computer Science Review*, 38:100288. DOI: 10.1016/j.cosrev.2020.100288.

Kumar, S., Arora, M., Chopra, S., *et al*. (2022). A re-

view of effort estimation in agile software development using machine learning techniques. In *2022 4th International Conference on Inventive Research in Computing Applications (ICIRCA)*, pages 416–422. IEEE. DOI: 10.1109/icirca54612.2022.9985542.

Malgonde, O. and Chari, K. (2019). An ensemble-based model for predicting agile software development effort. *Empirical Software Engineering*, 24:1017–1055. DOI: 10.1007/s10664-018-9647-0.

Marapelli, B., Carie, A., and Islam, S. M. (2020). Rnn-cnn model: A bi-directional long short-term memory deep learning network for story point estimation. In *2020 5th International Conference on Innovative Technologies in Intelligent Systems and Industrial Applications (CITISIA)*, pages 1–7. IEEE. DOI: 10.1109/CITISIA50690.2020.9371770.

Mikolov, T., Chen, K., Corrado, G., and Dean, J. (2013a). Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*. DOI: 10.48550/arxiv.1301.3781.

Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., and Dean, J. (2013b). Distributed representations of words and phrases and their compositionality. *Advances in neural information processing systems*, 26. DOI: 10.48550/arxiv.1310.4546.

Minku, L. L. and Yao, X. (2013). Ensembles and locality: Insight on improving software effort estimation. *Information and Software Technology*, 55(8):1512–1528. DOI: 10.1016/j.infsof.2012.09.012.

Moharreri, K., Sapre, A. V., Ramanathan, J., and Ramnath, R. (2016). Cost-effective supervised learning models for software effort estimation in agile environments. In *2016 IEEE 40th Annual computer software and applications conference (COMPSAC)*, volume 2, pages 135–140. IEEE. DOI: 10.1109/COMPSAC.2016.85.

Najm, A., Zakrani, A., and Marzak, A. (2019). Decision trees based software development effort estimation: A systematic mapping study. In *2019 International Conference of Computer Science and Renewable Energies (ICCSRE)*, pages 1–6. IEEE. DOI: 10.1109/iccsre.2019.8807544.

Naseem, U., Razzak, I., Khan, S. K., and Prasad, M. (2021). A comprehensive survey on word representation models: From classical to state-of-the-art word representation language models. *Transactions on Asian and Low-Resource Language Information Processing*, 20(5):1–35. DOI: 10.1145/3434237.

Nassif, A. B., Azzeh, M., Capretz, L. F., and Ho, D. (2013). A comparison between decision trees and decision tree forest models for software development effort estimation. In *2013 Third International Conference on Communications and Information Technology (ICCIT)*, pages 220–224. IEEE. DOI: 10.1109/ICCITechnology.2013.6579553.

Panda, A., Satapathy, S. M., and Rath, S. K. (2015). Empirical validation of neural network models for agile software effort estimation based on story points. *Procedia Computer Science*, 57:772–781. DOI: 10.1016/j.procs.2015.07.474.

Pennington, J., Socher, R., and Manning, C. D. (2014). Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543. DOI: 10.3115/v1/d14-1162.

Petersen, K., Feldt, R., Mujtaba, S., and Mattsson, M. (2008). Systematic mapping studies in software engineering. In *Proceedings of the 12th International Conference on Evaluation and Assessment in Software Engineering*, pages 68–77. BCS Learning & Development. DOI: 10.14236/ewic/EASE2008.8.

Petersen, K., Vakkalanka, S., and Kuzniarz, L. (2015). Guidelines for conducting systematic mapping studies in software engineering: An update. *Information and Software Technology*, 64:1–18. DOI: 10.1016/j.infsof.2015.03.007.

Phan, H. and Jannesari, A. (2022a). Heterogeneous graph neural networks for software effort estimation. In *Proceedings of the 16th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement*, pages 103–113. DOI: 10.1145/3544902.3546248.

Phan, H. and Jannesari, A. (2022b). Story point level classification by text level graph neural network. In *Proceedings of the 1st International Workshop on Natural Language-based Software Engineering*, pages 75–78. DOI: 10.1145/3528588.3528654.

Porru, S., Murgia, A., Demeyer, S., Marchesi, M., and Tonelli, R. (2016). Estimating story points from issue reports. In *Proceedings of the the 12th international conference on predictive models and data analytics in software engineering*, pages 1–10. DOI: 10.1145/2972958.2972959.

Pospieszny, P., Czarnacka-Chrobot, B., and Kobylinski, A. (2018). An effective approach for software project effort and duration estimation with machine learning algorithms. *Journal of Systems and Software*, 137:184–196. DOI: 10.1016/j.jss.2017.11.066.

Prasada Rao, C., Siva Kumar, P., Rama Sree, S., and Devi, J. (2018). An agile effort estimation based on story points using machine learning techniques. In *Proceedings of the Second International Conference on Computational Intelligence and Informatics: ICCII 2017*, pages 209–219. Springer. DOI: 10.1007/978-981-10-8228-3_20.

Praynlin, E. and Latha, P. (2013). Performance analysis of software effort estimation models using neural networks. *International Journal of Information Technology and Computer Science (IJITCS)*, 5(9):101–107. DOI: 10.5815/ijitcs.2013.09.11.

Pressman, R. S. (2011). Engenharia de software: uma abordagem profissional. 7ª edição. *Ed: McGraw Hill*. Book.

Ramchurreetoo, Y. and Hurbungs, V. (2022). A multiclass classification model to estimate agile user stories. In *2022 3rd International Conference on Next Generation Computing Applications (NextComp)*, pages 1–5. IEEE. DOI: 10.1109/NextComp55567.2022.9932190.

Ramessur, M. A. and Nagowah, S. D. (2021). A predictive model to estimate effort in a sprint using machine learning techniques. *International Journal of Information Technology*, 13(3):1101–1110. DOI: 10.1007/s41870-021-00669-z.

Rodríguez Sánchez, E., Vázquez Santacruz, E. F., and Cervantes Maceda, H. (2023). Effort and cost estimation using decision tree techniques and story points in agile software development. *Mathematics*, 11(6):1477. DOI:

10.3390/math11061477.

Russell, S. (2016). *Artificial Intelligence: A Modern Approach, eBook, Global Edition*. Pearson Education, Limited. Book.

Sánchez, E. R., Maceda, H. C., and Santacruz, E. V. (2022). Software effort estimation for agile software development using a strategy based on k-nearest neighbors algorithm. In *2022 IEEE Mexican International Conference on Computer Science (ENC)*, pages 1–6. IEEE. DOI: 10.1109/ENC56672.2022.9882947.

Santos, I., Nedjah, N., and de Macedo Mourelle, L. (2017). Sentiment analysis using convolutional neural network with fasttext embeddings. In *2017 IEEE Latin American conference on computational intelligence (LA-CCI)*, pages 1–5. IEEE. DOI: 10.1109/LA-CCI.2017.8285683.

Sarro, F., Moussa, R., Petrozziello, A., and Harman, M. (2020). Learning from mistakes: Machine learning enhanced human expert effort estimates. *IEEE Transactions on Software Engineering*, 48(6):1868–1882. DOI: 10.1109/TSE.2020.3040793.

Satapathy, S. M. and Rath, S. K. (2017). 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. DOI: 10.1007/s11334-017-0288-z.

Scott, E. and Pfahl, D. (2018). Using developers' features to estimate story points. In *Proceedings of the 2018 International Conference on Software and System Process*, pages 106–110. DOI: 10.1145/3202710.3203160.

Sharma, A. and Chaudhary, N. (2020). Linear regression model for agile software development effort estimation. In *2020 5th IEEE International Conference on Recent Advances and Innovations in Engineering (ICRAIE)*, pages 1–4. IEEE. DOI: 10.1109/ICRAIE51050.2020.9358309.

Shepperd, M. (2007). Software project economics: a roadmap. In *Future of Software Engineering (FOSE'07)*, pages 304–315. IEEE. DOI: 10.1109/FOSE.2007.23.

Sinclair, J. and Wynne, M. (2004). Developing linguistic corpora: A guide to good practice. *Ahds literature, languages and linguistics*. Available at:`https://users.ox.ac.uk/~martinw/dlc/`.

Singh, A. and Kumar, M. (2020). Comparative analysis on prediction of software effort estimation using machine learning techniques. In *Proceedings of the International Conference on Innovative Computing & Communications (ICICC)*. DOI: 10.2139/ssrn.3565813.

Soares, R. G. (2018). Effort estimation via text classification and autoencoders. In *2018 International Joint Conference on Neural Networks (IJCNN)*, pages 01–08. IEEE. DOI: 10.1109/IJCNN.2018.8489030.

SOMMERVILLE, I. (2004). *Engenharia de Software.* São Paulo: Pearson Addison-Wesley. Book.

Sommerville, I. (2011). Engenharia de software-8ª edição 2007. *Ed Person Education*. Book.

Sree, S., Rao, P., and Mounika, M. (2017). An early-stage software effort estimation in agile methodology based on user stories using machine learning techniques. *International Journal for Research in Applied Science Engineering Technology (IJRASET)*, 5. Available at:`https://www.ijraset.com/fileserve.php?FID=11793`.

Srinivasan, K. and Fisher, D. (1995). Machine learning approaches to estimating software development effort. *IEEE Transactions on Software Engineering*, 21(2):126–137. DOI: 10.1109/32.345828.

Sudarmaningtyas, P. and Mohamed, R. (2021). A review article on software effort estimation in agile methodology. *Pertanika Journal of Science & Technology*, 29(2):837–861. DOI: 10.47836/pjst.29.2.08.

Suresh, Y. (2022). Effective ann model based on neuroevolution mechanism for realistic software estimates in the early phase of software development. *International Journal of Advanced Computer Science and Applications*, 13(2). DOI: 10.14569/IJACSA.2022.0130223.

Suresh Kumar, P., Behera, H., Nayak, J., and Naik, B. (2022). A pragmatic ensemble learning approach for effective software effort estimation. *Innovations in Systems and Software Engineering*, 18(2):283–299. DOI: 10.1007/s11334-020-00379-y.

Tawosi, V., Moussa, R., and Sarro, F. (2022). Agile effort estimation: Have we solved the problem yet? insights from a replication study. *IEEE Transactions on Software Engineering*, 49(4):2677–2697. DOI: 10.1109/TSE.2022.3228739.

Turic, M., Celar, S., Dragicevic, S., and Vickovic, L. (2023). Advanced bayesian network for task effort estimation in agile software development. *Applied Sciences*, 13(16):9465. DOI: 10.3390/app13169465.

Van Houdt, G., Mosquera, C., and Nápoles, G. (2020). A review on the long short-term memory model. *Artificial Intelligence Review*, 53:5929–5955. DOI: 10.1007/s10462-020-09838-1.

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. DOI: 10.65215/nxvz2v36.

Wang, C. and Zhang, F. (2022). The performance of improved xlnet on text classification. In *Third International Conference on Artificial Intelligence and Electromechanical Automation (AIEA 2022)*, volume 12329, pages 154–159. SPIE. DOI: 10.1117/12.2646785.

Wazlawick, R. (2010). *Análise e Projeto de Sistemas de Informação Orientados*, volume 2. Elsevier Brasil. Book.

Wohlin, C. (2014). Guidelines for snowballing in systematic literature studies and a replication in software engineering. In *Proceedings of the 18th international conference on evaluation and assessment in software engineering*, pages 1–10. DOI: 10.1145/2601248.2601268.

Wohlin, C., Runeson, P., Host, M., Ohlsson, M. C., Regnell, B., and Wesslen, A. (2012). *Experimentation in Software Engineering: An Introduction*. Springer-Verlag Berlin Heidelberg, 1st. edition. DOI: 10.1007/978-3-662-69306-3.

Yang, Z. (2019). Xlnet: Generalized autoregressive pretraining for language understanding. *arXiv preprint arXiv:1906.08237*. DOI: 10.1145/3491065.

Yousef, Q. M., Alshaer, Y. A., and Alhammad, N. K. (2017). Dragonfly estimator: a hybrid software projects' efforts estimation model using artificial

neural network and dragonfly algorithm. *Int. J. Comput. Sci. Netw. Secur*, 17(9):108–120. Available at:https://www.researchgate.net/publication/352135234_Dragonfly_Estimator_A_Hybrid_Software_Projects%27_Efforts_Estimation_Model_using_Artificial_Neural_Network_and_Dragonfly_Algorithm.

Zakrani, A., Najm, A., and Marzak, A. (2018). Support vector regression based on grid-search method for agile software effort prediction. In *2018 IEEE 5th International Congress on Information Science and Technology (CiSt)*, pages 1–6. IEEE. DOI: 10.1109/CIST.2018.8596370.

Zhang, D., Xu, H., Su, Z., and Xu, Y. (2015). Chinese comments sentiment classification based on word2vec and svmperf. *Expert Systems with Applications*, 42(4):1857–1863. DOI: 10.1016/j.eswa.2014.09.011.