# GPT2SP: A Transformer-Based Agile Story Point Estimation Approach

Michael Fu, *Student Member, IEEE* and Chakkrit Tantithamthavorn, *Member, IEEE*

**Abstract**—Story point estimation is a task to estimate the overall effort required to fully implement a product backlog item. Various estimation approaches (e.g., Planning Poker, Analogy, and expert judgment) are widely-used, yet they are still inaccurate and may be subjective, leading to ineffective sprint planning. Recent work proposed Deep-SE, a deep learning-based Agile story point estimation approach, yet it is still inaccurate, not transferable to other projects, and not interpretable. In this paper, we propose GPT2SP, a Transformer-based Agile Story Point Estimation approach. Our GPT2SP employs a GPT-2 pre-trained language model with a GPT-2 Transformer-based architecture, allowing our GPT2SP models to better capture the relationship among words while considering the context surrounding a given word and its position in the sequence and be transferable to other projects, while being interpretable. Through an extensive evaluation on 23,313 issues that span across 16 open-source software projects with 10 existing baseline approaches for within- and cross-project scenarios, our results show that our GPT2SP approach achieves a median MAE of 1.16, which is (1) 34%-57% more accurate than existing baseline approaches for within-project estimations; (2) 39%-49% more accurate than existing baseline approaches for cross-project estimations. The ablation study also shows that the GPT-2 architecture used in our approach substantially improves Deep-SE by 6%-47%, highlighting the significant advancement of the AI for Agile story point estimation. Finally, we develop a proof-of-concept tool to help practitioners better understand the most important words that contributed to the story point estimation of the given issue with the best supporting examples from past estimates. Our survey study with 16 Agile practitioners shows that the story point estimation task is perceived as an extremely challenging task. In addition, our AI-based story point estimation with explanations is perceived as more useful and trustworthy than without explanations, highlighting the practical need of our Explainable AI-based story point estimation approach.

**Index Terms**—Agile story point estimation, AI for SE, explainable AI

## 1 Introduction

"*Good estimation helps product owners optimize for efficiency and impact."–an Atlassian manager.* [1]

Story point estimation is one of the most difficult task in Agile. Story points are units of measure for expressing an estimate of the overall effort required to fully implement a product backlog item. Typically, story points are estimated by the team using various approaches based on team consensus, e.g., Planning Poker, Analogy, and expert judgment, by considering the amount of work, complexity, risk, and uncertainty. However, Usman *et al.* [53] point out that the subjective estimation based on domain experts may introduce bias. Hence, such inaccurate story point estimation (i.e., overestimate/underestimate) could lead to ineffective sprint planning, resulting in idle developer time, cost overruns, or project failure.

Thus, various Artificial Intelligence (AI) and Machine Learning (ML) have been used for story point estimations [7], [11], [36]. For example, Porru [36] used Bag-of-Words (BoW) features with various machine learning techniques (e.g., support vector machine). However, such Bag-of-Words (BoW) features needed to be handcrafted, which is time-consuming. Recently, Choetkiertikul *et al.* [7] proposes a Deep-SE approach, which is an end-to-end deep learning approach for agile story point estimations. Deep-SE starts from building a pre-trained language model for generating vector representations. Then, Deep-SE employs Long Short-Term Memory (LSTM) with Recurrent Highway Network (RHWN) to automatically learn the distributed representation of words in order to estimate story points. However, the Deep-SE approach has the following three limitations.

- First, Deep-SE builds a pre-trained language model from their own story point datasets for each project. Such the project-specific pre-training is very time consuming (i.e., Deep-SE takes 2-7 hours to build a pre-trained model for each project), limiting the understanding of the language models to the known vocabs within the trained project, which is not transferable to other projects (as demonstrated by the inaccurate cross-project estimations of Deep-SE).
- Second, Deep-SE employs LSTM to learn the embedding vectors generated from the pre-trained models. Due to the sequential nature of LSTM, the unidirectional

- *The authors are with the Faculty of Information Technology, Monash University, Melbourne, VIC 3800, Australia. E-mail: {yeh.fu, chakkrit}@monash.edu.*

*Manuscript received 1 September 2021; revised 7 March 2022; accepted 7 March 2022. Date of publication 10 March 2022; date of current version 13 February 2023.*
*This work of Chakkrit Tantithamthavorn was supported in part by Australian Research Council's Discovery Early Career Researcher Award (DECRA) under Grant DE200100941. The work of Michael Fu was supported in part by the Faculty of Information Technology Postgraduate Research Scholarship from Monash University (Clayton).*
*This work involved human subjects or animals in its research. Approval of all ethical and experimental procedures and protocols was granted by Monash University Human Research Ethics Committee (MUHREC) (Application No. 31180).*
*(Corresponding author: Chakkrit Tantithamthavorn.)*
*Recommended for acceptance by E. Murphy-Hill.*
*Digital Object Identifier no. 10.1109/TSE.2022.3158252*

LSTM only processes words in one direction (i.e., from left to right). Hence, such LSTM networks are not able to capture the global word dependencies from the current words to the past words (i.e., from right to left) in a sequence, limiting the ability to capture the rich semantic meanings and the relationship between the words in the issue and the story points.

- Finally, the Deep-SE architecture is not yet interpretable, preventing researchers and practitioners to understand what words contribute the most to the predictions. Such a lack of interpretability often hinders the adoption in software development practices [7], [9], [48], [49].

In this paper, we propose GPT2SP, a Transformer-based Agile Story Point Estimation approach to address the three limitations of Deep-SE. First, instead of using word-level tokenization, we employ a byte-pair-encoding subword tokenization to split rare words into subword units, reducing the vocabulary size by 73% (from 186,625 vocabs to 50,257 vocabs). Second, instead of using project-specific pre-trained models, we employed a GPT-2 language model, allowing our GPT2SP approach to generate more meaningful vectors for any projects. Third, instead of using LSTM networks, we employ the GPT-2 architecture [38] with a masked multi-head self-attention mechanism [54], allowing our GPT2SP approach to better capture the relationship among words while considering the context surrounding a given word and its position in the sequence.

Finally, we evaluate our approach using Mean Absolute Error and compare with other existing nine baseline approaches (i.e., Deep-SE [7], LSTM+RF, LSTM+SVM, LSTM +ATLM, LSTM+LR, Doc2Vec+RF, BoW+RF, Mean, and Median) for within-project scenarios. We then compare our approach with Deep-SE and Analogy-based Estimation (ABE0) for cross-project scenarios. Through an extensive evaluation of our approach on 23,313 issues that span across 16 open-source software projects, we address the following three research questions:

(RQ1) *Does our GPT2SP outperform Deep-SE for within-project scenarios?*

Results. We find that our GPT2SPachieves a median MAE of 1.16, which is 34%-57% significantly more accurate than the existing nine baseline approaches. The ScottKnott ESD test also confirms that our GPT2SP is the only approach that appeared in Rank-1, indicating that our GPT2SP models statistically outperform other existing baseline approaches with non-negligible difference for within-project evaluations.

(RQ2) *Does our GPT2SP outperform Deep-SE for cross-project scenarios?*

Results. We find that, regardless of the used training set from within-repository or cross-repository, our GPT2SP is still the most accurate for cross-project estimations when compared with the Deep-SE and ABE0 (Analogy-based Estimation). In particular, our GPT2SPachieves a median MAE of 2.14, which is more accurate than than Deep-SE by 39% and ABE0 by 49% for cross-repository evaluations, highlighting the benefits of using GPT-2 language models to learn the distributed representations of words in a broader context than the Deep-SE language models.

(RQ3) *What are the contributions of the components of GPT2SP?*

Results. We find that the Transformer architecture used in our GPT2SP substantially improves the MAE of Deep-SE by 6% to 47%, highlighting the substantial benefits of using the Transformer architecture for Agile story point estimation. On the other hands, the choice of subword tokenization algorithms have little impact on the MAE. Nevertheless, BPE is still the best subword toknization algorithm for our GPT2SP.

These results lead us to conclude that our GPT2SP is the most accurate agile story point estimation techniques when compared to the existing ten baseline approaches (including Deep-SE [7]), highlighting the significant advancement of the AI for Agile story point estimation.

Survey Study. To help practitioners better understand the story point estimation from our GPT2SP models, we develop a proof-of-concept web-based Agile story point estimation tool. Taking an issue as input, our GPT2SP tool estimates a story point; highlights the most important word that contributed to the story point estimation of the given issue; and provides supporting examples from the training set of the same project. Our survey study with 16 Agile practitioners indicates that the story point estimation task is perceived as an extremely challenging task. In addition, our AI-based story point estimation with explanations is perceived as more useful and trustworthy than without explanations, highlighting the practical need of our Explainable AI-based story point estimation. Our model inspection analysis shows that our GPT2SP can uncover 20% supporting examples in the training that have the same important token and the same story points, indicating that our GPT2SP is able to learn the relationship between issues and story points from the past estimates in order to correctly estimate a story point for a given issue.

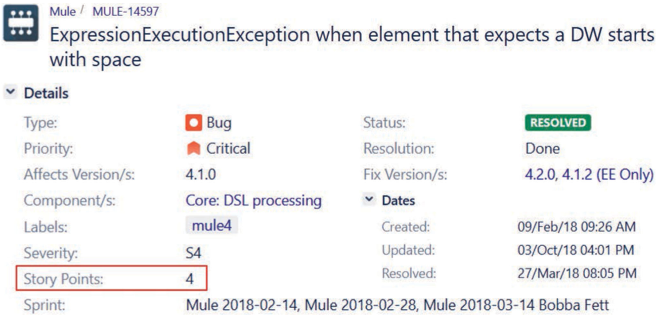
*Novelty & Contributions*. To the best of our knowledge, this paper is the first to present:

- A Transformer-based Agile story point estimation approach (GPT2SP).
- An extensive evaluation of 23,313 issues with existing nine baseline approaches (including Deep-SE [7]) for both within-project and cross-project evaluation scenarios.
- An ablation study to quantify the contributions of the two components (i.e., BPE and Transformer) used in our GPT2SP.
- A proof-of-concept web-based Agile story point estimation tool (URL: https://share.streamlit.io/awsm-research/gpt2sp_webapp/main/app.py).

Open Science. To facilitate future work, we publish the studied dataset, scripts, and experimental results (e.g., raw predictions) in GitHub (https://github.com/awsm-research/gpt2sp).

## 2 Background

*Agile* is an iterative development process which consists of four main steps, i.e., (1) product backlog refinement, (2) sprint planning, (3) sprint executing, and (4) sprint delivering. At


Mule / MULE-14597

ExpressionExecutionException when element that expects a DW starts with space

Details

| | | | |
|---|---|---|---|
| Type: | Bug | Status: | RESOLVED |
| Priority: | Critical | Resolution: | Done |
| Affects Version/s: | 4.1.0 | Fix Version/s: | 4.2.0, 4.1.2 (EE Only) |
| Component/s: | Core: DSL processing | Dates | |
| Labels: | mule4 | Created: | 09/Feb/18 09:26 AM |
| Severity: | S4 | Updated: | 03/Oct/18 04:01 PM |
| Story Points: | 4 | Resolved: | 27/Mar/18 08:05 PM |
| Sprint: | Mule 2018-02-14, Mule 2018-02-28, Mule 2018-03-14 Bobba Fett | | |

Fig. 1. An example JIRA issue with an associated story point.

the beginning, product owners collect software requirements from customer representatives in the form of a list of work items to develop the software (aka a product backlog). In the first step of the product backlog refinement, the team reviews and refines the work items in the product backlog. Since some work items may be large (e.g., Epics which describes a high-level overview of a feature), the team performs work breakdowns by creating a set of smaller work items (aka stories or work items). Then, the team estimates the effort required to complete work items and prioritizes them. Finally, the team performs a sprint planning, defines the sprint goal, determines the team capacity, selects work items according to their capacity into a sprint backlog, and executes the sprint in a short iteration in order to deliver a working product [3].

*Story Points (SP)* are estimates of relative effort of a work item. Work items are often called as *issue* in JIRA issue tracking systems. Typically, story points are estimated by the team using various approaches based on team consensus, e.g., Planning Poker, Analogy, and expert judgment [52], by considering the amount of work, complexity, risk, and uncertainty. However, Usman *et al.* [53] point out that the subjective estimation based on domain experts' experience may introduce bias. Hence, such inaccurate story point estimation (i.e., overestimate/underestimate could lead to ineffective sprint planning, resulting in idle developer time, loss of productivity, cost overruns, project failure, customer dissatisfaction, and loss of business.

Recently, Choetkiertikul *et al.* [7] propose *Deep-SE*, an end-to-end DL-based approach for Agile Story Point Estimation. They evaluate the model on both within-project and cross-project scenarios where the training data and testing data are from two different projects.

Deep-SE is evaluated using 23,313 issues (aka work items) from 16 open source projects that use JIRA issue tracking. Each issue typically has a title and a description, and a story point associated with that issue. Fig. 1 provides an example JIRA issue with an associated story point. Given an issue, Deep-SE automatically learns semantic features which represent the meaning of issues using a deep learning architecture, i.e., Long Short-Term Memory (LSTM) and Recurrent Highway Network (RHWN). Their experiments show that Deep-SE achieves an average Mean Absolute Error of 2.08, which outperforms other ML-based approaches (i.e., LSTM+RF, BoW+RF, Doc2Vec+RF, TFIDF+SVM[36]) and simple baselines. Deep-SE consists of four steps:

*Step 1: Word-Embedding.* Words in issue reports written in natural languages often have their own semantic meanings, yet they are hard to represent.

Thus, Deep-SE learns the distributed representation of words by building a pre-trained language model in an unsupervised manner using an unlabeled corpora of domain specific data (e.g., issue reports). To do so, they first combine the title and description of an issue report into a single textual document where the title is followed by the description. Each word in an issue report is represented as a low dimensional, continuous and real-valued vector.

*Step 2: Document Representation using LSTM.* Since a document consists of a sequence of words, Deep-SE employs a Long Short-Term Memory (LSTM) unit, which is a special variant of Recurrent Neural Network (RNN). Then, the LSTM generates a sequence of state vectors, which are then pooled to form a document-level vector.

*Step 3: Deep Representation using RHWN.* To prevent overfitting, Deep-SE employs a Recurrent Highway Network (RHWN) to transform the document vector multiple times before generating a final vector which represents the document.

*Step 4: Regressor.* Finally, Deep-SE takes the document vector as input into a linear activation function in a feedforward neural network to estimate a story point.

## 3 The GPT2SPArchitecture

In this section, we present the GPT2SP architecture, which is a GPT-2 Transformer-based Agile Story Point Estimator. Fig. 2 presents an overview of our GPT2SP Architecture.

Overview. Given an issue report, in Step ①, we perform subword tokentization using a byte-pairs encoding (BPE) approach based on a GPT-2 pre-trained language model in order to produce subword-tokenized issues (i.e., a list of subwords for each issue). In Step ②, we build a GPT2SP model based on a GPT-2 architecture. For each subword-tokenized issue, in Step ②a, GPT2SPperforms a word & positional encoding in order to generate an embedding vector of each word and its position in the issue. Then, in Step ②b, the vector is fed into the GPT-2 architecture, which is a stack of 12 Transformer decoder blocks. Each decoder consists of a masked multi-head self-attention and a feed forward neural network. Then, the output vector is fed into a Multi-Layer Perceptron in order to estimate the story point of the given issue. Below, we describe the details in each step.

### 3.1 Subword Tokenization

Tokenization is an important step in natural language processing, aiming to breaking unstructured data and natural language text into chunks of information that can be considered as discrete elements. There exist three main granularity levels of tokenization approaches, i.e, words, subwords, and characters. In the prior work [7], Deep-SE used word-level tokenization (i.e., breaking a sentence into words). However, the word-level tokenization will produce an extremely large corpus of vocabularies (i.e, there are 185,625 unique words in the Deep-SE corpus), limiting the ability of the Deep-SE approach to generate meaningful representation of words. With this limitation, Deep-SE can only generate a vector representation of vocab in the training dataset, but not for a new vocab that hasn't been learned before. Although character-level tokenization can address the open vocabulary problem, each character lacks of meaning, and

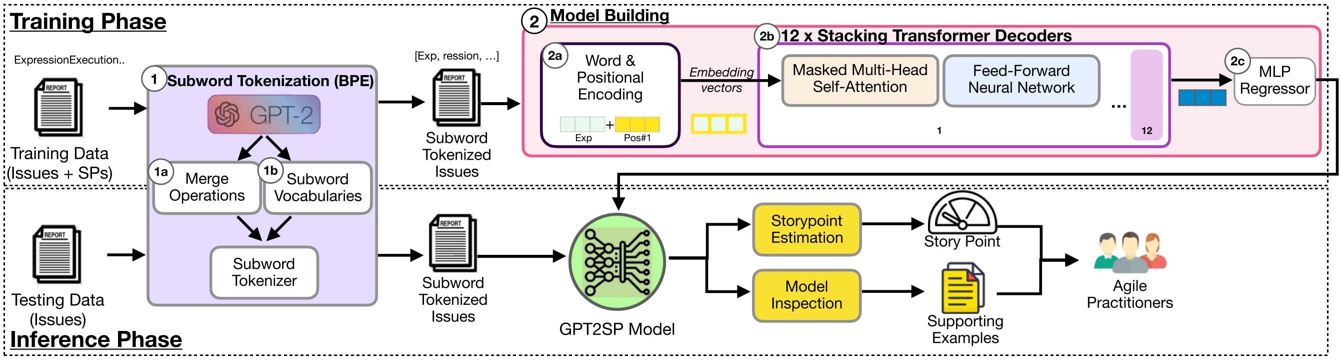


Fig. 2. An overview architecture of our GPT2SP.

increases the input computation, limits network choices due to excessively large input sequences.

To address this limitation, we perform a subword tokenization using a Byte-Pair-Encoding (BPE) algorithm [12], [44]. Subword tokenization will preserve the common words (i.e., will not split the common words into smaller subwords), but only split rare words into meaningful subwords. For example, the title of the given issue (see Fig. 1) is "ExpressionExecutionException when element ...". Thus, this title will be split into a list of subwords, i.e., [”Exp”, ”ression”, ”Exec”, ”ution”, ”Exception”, ”when”, ”element”, ...]. Since ’ExpressionExecutionException’ is a rare word, it is split into four subwords. On the other hands, other common words are preserved to the original form (e.g., when, element). Below, we briefly describe the BPE algorithm [44].

BPE performs two main steps: (1a) generating merge operations, and (1b) applying merge operations based on the subword vocabularies. Merge operations are used to determine how a word should be split. To generate merge operations, in Step (1a), BPE will first split all words into characters sequences. Then, BPE generates a merge operation by identifying the most frequent symbol pair (e.g., the pair of two consecutive characters) that should be merged into a new symbol.

In this paper, we use the merge operations and the subword vocabularies generated from the GPT-2 corpus (i.e., a massive 40GB of 8 million web pages) for pre-training the GPT-2 language model. Different from prior work [7], BPE used by our GPT2SP approach substantially reduces the vocab size by 73% (i.e., from 185,625 vocabs to 50,257 vocabs).

## 3.2 Model Building

Issues are written in natural language, which heavily relies on context and the position of each word in a sentence. Hence, it is important to capture the word dependencies within the issues, especially, for learning issues for estimating story points. Let’s consider the example issue in Fig. 1 with the title “ExpressionExecutionException when element that expects a DW starts with space”. This issue title is about an exception was thrown when something unexpected happens. In particular, the word “element” is related to the “Exception”, while the word “expect” is related to “DW”, and the word “start” is related to “space”. Yet, the understanding of such complex and context dependent issue titles is a challenging problem of AI for story point estimations.

Previously, Deep-SE builds a pre-trained language model for each project to generate a vector representation of each word, limiting the understanding of the language models to the known vocabs within the trained project and limiting the transferability of Deep-SE that learned from one project to other projects. Then, Deep-SE employs LSTM to learn the vector representation. Due to the sequential nature of LSTM, the unidirectional LSTM only processes words in one direction (i.e., from left to right). Hence, it is not able to capture the global word dependencies from the current words to the past words (i.e., from right to left) in a sequence, limiting the ability to capture the rich semantic meanings and the relationship between the words in the issue and the story points.

To address these limitations of Deep-SE [7], we use a GPT-2 language model to generate a vector representation of each token. The GPT-2 language model is a foundation model that is pre-trained on broad data at scale which can be adapted (e.g., fine-tuned) to a wide range of downstream tasks [6]. The GPT-2 language model is pre-trained from a massive 40GB of 8 million web pages in an unsupervised fashion (i.e., without labels), allowing our GPT2SP approach to better capture the relationship among words and generate more meaningful vectors for any projects. Due to the nature of the foundation model, our GPT2SP approach is transferable to other projects without the need to build a pre-trained language model for every project like Deep-SE (i.e., Deep-SE takes 2-7 hours to build a pre-training model for each project).

Generally, our GPT2SP model building consists of 3 parts: word & positional encoding, a stack of 12 decoder-only Transformer blocks, and a fully-connected layer of multi-layer perceptron.

(2a) *Word & Positional Encoding*. The goal of this step is to generate encoding vectors that capture the semantic meaning of the word and its position in the input sequence. To do so, for each word in a subword-tokenized issue, we generate two vectors: (1) a word encoding vector to represent the meaningful relationship between a given word and the other tokens and (2) the positional encoding vector to represent the position of a given token in the input sequence. The token encoding vectors are generated according to the word embedding matrix $\mathrm{W}_{te}^{|V|\times d}$ where $|V|$ is the vocabulary size and $d$ is an embedding size. The positional encoding vectors are generated according to the positional embedding matrix $\mathrm{W}_{pe}^{c\times d}$ where $c$ is the context size and $d$ is the embedding size. Then, both the word encoding vector and the positional encoding vector are concatenated in order to produce an encoding vector into our GPT2SP.

TABLE 1
The Descriptive Statistics of the Story Point Datasets Provided by Choetkiertikul *et al.* [7]

| Repository | Project | #Issues | SP$_{min}$ | SP$_{max}$ | SP$_{mean}$ | SP$_{median}$ | SP$_{var}$ | SP$_{std}$ |
|---|---|---|---|---|---|---|---|---|
| Apache | Mesos | 1,680 | 1 | 40 | 3.09 | 3 | 5.87 | 2.42 |
| | Usergrid | 482 | 1 | 8 | 2.85 | 3 | 1.97 | 1.4 |
| Appcelerator | Appcelerator Studio | 2,919 | 1 | 40 | 5.64 | 5 | 11.07 | 3.33 |
| | Aptana Studio | 829 | 1 | 40 | 8.02 | 8 | 35.46 | 5.95 |
| | Titanium SDK / CLI | 2,251 | 1 | 34 | 6.32 | 5 | 25.97 | 5.1 |
| Dura Space | DuraCloud | 666 | 1 | 16 | 2.13 | 1 | 4.12 | 2.03 |
| Atlassian | Bamboo | 521 | 1 | 20 | 2.42 | 2 | 4.6 | 2.14 |
| | Clover | 384 | 1 | 40 | 4.59 | 2 | 42.95 | 6.55 |
| | JIRA Software | 352 | 1 | 20 | 4.43 | 3 | 12.35 | 3.51 |
| Moodle | Moodle | 1,166 | 1 | 100 | 15.54 | 8 | 468.53 | 21.65 |
| Lsstcorp | Data Management | 4,667 | 1 | 100 | 9.57 | 4 | 275.71 | 16.61 |
| Mulesoft | Mule | 889 | 1 | 21 | 5.08 | 5 | 12.24 | 3.5 |
| | Mule Studio | 732 | 1 | 34 | 6.4 | 5 | 29.01 | 5.39 |
| Spring | Spring XD | 3,526 | 1 | 40 | 3.7 | 3 | 10.42 | 3.23 |
| Talendforge | Talend Data Quality | 1,381 | 1 | 40 | 5.92 | 5 | 26.96 | 5.19 |
| | Talend ESB | 868 | 1 | 13 | 2.16 | 2 | 2.24 | 1.5 |
| | TOTAL | 23,313 | | | | | | |

2b *A Stack of 12 Decoder-only Transformer Blocks.* In this Step, the encoding vectors are fed into a stack of 12 decoder-only Transformer blocks (i.e., the GPT-2 architecture [38]). Each decoder block consists of two components, i.e., a masked multi-head self-attention [54] and a fully-connected feed-forward neural network. Below, we briefly describe the masked multi-head self-attention and the feed-forward neural network.

The masked multi-head self-attention [38] is used to compute an attention weight of each word, producing an attention vector which will be used in the Decoder block to indicate which words that the Transformer model should pay attention to. Generally, the masked self-attention mechanism is used to obtain global dependencies where the (*self-attention*) weights are indicative of how each word of the sequence is influenced by all the other words in the sequence without attending the subsequent positions (*masked*), allowing our GPT2SP approach to capture dependencies between every word which leads to more meaningful representation.

The self-attention mechanism [54] employs a concept of information retrieval (i.e., computing the relevant scores of other words in the sequence based on a given word). The self-attention mechanism consists of three main components, i.e., Query ($Q$), Key ($K$), and Value ($V$). The Query is a representation of the current word used to score against all the other words (using their Keys), while the Key vectors are labels for all the other words in the sequence that are used to search for relevant words, and the Value vectors are vectors after multiplying them by their associated score (i.e., dot product followed by softmax) using the following calculation: Attention$(Q, K, V) = \mathrm{softmax}(\frac{QK}{\sqrt{d_k}})V$.

A *multi-head* mechanism is used to jointly attend to parts of the sequence differently in parallel in order to capture richer semantic meanings of the input sequence. The multi-head mechanism can be described as the following calculation: MultiHead$(Q, K, V) =$ Concat$(\mathrm{head}_1, \ldots, \mathrm{head}_h)W^O$, where head$_i$ = Attention($QW_i^Q$, KW$_i^K$, VW$_i^V$), where $W^O$ is used to linearly project to the expected dimension before being sent to a fully connected neural network. Finally, the vectors are fed into a fully-connected feed-forward neural network.

2c*Multi-Layer Perceptron Regressor.* Since GPT-2 is mainly designed for text generation tasks (e.g., Text→Text), the original GPT-2 architecture is not directly applicable to our story point estimation task (i.e., Text→Number). Instead of using a single linear layer to predict the probabilities of the next word, we modify the last layer of GPT-2 to a 3-layer Multi-Layer Perceptron (MLP) regressor to estimate the story point for a given input issue. The predicted function is applied to map the input features to the output target (i.e., story point), which can be expressed as follows: $\hat{y}(X) = b + \sum_{i=1}^{M} w_i$, where $M$ is the number of hidden nodes, $b$ is the bias of the output node, and $w_i$ represents the connection weights of output node to $i^{\mathrm{th}}$ node in the hidden layer.

## 4 Experimental Setup

In this section, we present the motivation of our three research questions, our studied datasets, our model implementation, our hyperparameter settings, our evaluation measure, and our statistical analysis.

### 4.1 Research Questions

To evaluate our GPT2SP, we formulate the following three research questions.

*(RQ1) Does our GPT2SP outperform Deep-SE for within-project scenarios?* Story point estimation is known to be specific to teams and projects. Choetkiertikul *et al.* [7] propose Deep-SE, a state-of-the-art deep learning approach for story point estimation, yet their approach is still inaccurate. Thus, we propose our GPT2SPto address the limitations of Deep-SE. Hence, we formulate this RQ to evaluate whether GPT2SPoutperforms Deep-SE for within-project estimation scenarios.

*(RQ2) Does our GPT2SP outperform Deep-SE for cross-project scenarios?* At the beginning of new Agile software development projects, story point estimation may be a difficult task for Agile practitioners due to lack of shared knowledge. Similarly, AI/ML approaches for story point estimation may also

be not accurate due to lack of training data. One common approach to address this issue is to perform a cross-project estimation, i.e., the model is trained on one source project, while applying it to another target project. Hence, we formulate this RQ to evaluate whether GPT2SPoutperforms Deep-SE for cross-project estimation scenarios.

*(RQ3) What are the contributions of the components of GPT2SP?* The architecture of our GPT2SPconsists of 2 major components ($\text{BPE}_{\text{GPT2}}$+GPT2), which is different from the Deep-SE architecture (WordEmb+LSTM+RHWN). Yet, little is known about how does each component of our approach contribute to the story point estimations. Hence, we formulate this RQ to empirically evaluate the contribution of each component of our GPT2SP.

### 4.2 Studied Datasets

To establish a fair comparison of our GPT2SP with the state-of-the-art Deep-SE approach, we use the same benchmark datasets provided by Choetkiertikul *et al.* [7]. They collect the dataset from JIRA, one of the few widely-used issue tracking systems that supports agile software development including the story point estimation with its JIRA Agile plugin. For each project, issues and related key information (i.e., issue ID, title, descriptions, and story points) are collected by Choetkiertikul *et al.*through JIRA REST API up until August 8, 2016. Table 1 describes the statistic of the datasets. Ultimately, we conduct our experiments with 23,313 issues from 16 different projects.

### 4.3 Model Implementation

To build our GPT2SP model, we mainly use two Python libraries, i.e., Transformers developed by HuggingFace [56] and PyTorch [30]. The Transformers library provides API access to the Transformer-based model architectures, while the PyTorch library supports the computation during the training (e.g., back-propagation and parameter optimization). To construct the GPT2SP models, we use the GPT-2 tokenizer based on the GPT-2 corpus provided by Transformers and build our custom GPT-2 architecture on top of the pre-trained GPT-2 model [39]. We use the training set to build GPT2SPmodels using an NVIDIA RTX 3090 graphic card. The model training time varies from 3 to 39 seconds for each project (see Table 4 in Appendix, which can be found on the Computer Society Digital Library at http://doi.ieeecomputersociety.org/10.1109/TSE.2022.3158252). Similar to any deep learning models, GPT-2 model involves various hyper-parameters. Thus, the learning process is achieved through the optimization process of finding the best set of hyper-parameters $\theta$ that minimizes the loss function $L(\theta)$ of the validation set (not the testing set). The L1 loss function $L(\theta)$ is used to minimize the error of our GPT2SP models, which is the sum of the all the absolute differences between the true value and the predicted value ($L(\theta) = \Sigma_{i=1}^{n}|y_{true} - y_{predicted}|$). We use an AdamW optimizer [26] with a learning rate (LR) of $5e-4$ to prevent over-fitting in the training process. The AdamW optimizer is a enhanced version of the Adam optimizer [21] with weight decay regularization, which is used to minimize the loss function, since it has been shown to be computationally efficient and require low memory consumption. Finally, we also use back-propagation, a simple implementation of the chain rule of partial derivatives, to efficiently update the parameters during the training process.

### 4.4 Hyper-Parameter Settings

Four different versions of GPT-2 have been proposed [38], [39]: small, medium, large, and xl. These variants differ in terms of the number of layers, the number of hidden states, the number of heads, and the number of total parameters. For example, the GPT-$2_{\text{small}}$ has 117 million parameters, while the GPT-$2_{\text{xl}}$ has 1.5 billion parameters. Prior works point out that the complex variants of Transformer models may increase the accuracy, the training complexity also increases as well due to the increasing number of parameters [28]. Thus, we expect that the results achieved in our study to be a lower bound for the performance of a GPT-2-based models. To avoid any excessive computation, we use the GPT-$2_{\text{small}}$, the smallest version of GPT-2 with the default hyper parameter settings, i.e., 12-layer, 768-hidden, 12-heads, 117M parameters. The GPT-$2_{\text{small}}$ is pre-trained from a vocabulary size of 50,257. Each word is embedded into an encoding vector size of 768. Each vector is fed into the GPT-$2_{\text{small}}$ architecture with a stack of 12 blocks of Transformer decoders (i.e., 12-layer). Each decoder block is identical with the masked multi-head self-attention mechanism and the feed-forward neural network. For the masked multi-head self-attention mechanism, the key and value matrices of all attention mechanisms have an inner dimension ($d_{kv}$) of 64. Thus, after the concatenation step, the size of the attention vectors will be 768 (64×12 heads). Then, these vectors are fed to the fully-connected feed-forward neural networks.

### 4.5 Evaluation Measure

We use Mean Absolute Error (MAE) as our evaluation measure for both within-project and cross-project experimental scenarios, which is also used to evaluate DeepSE by Choetkiertikul *et al.* [7]. MAE measures the average magnitude of the errors in a set of forecasts without considering their direction. Noted that the goal of this paper is to advance the state-of-the-art Deep-SE (not random guessing). Thus, other measures (e.g., MdAE, MMRE, SA) are not selected, since MdAE (Median Absolute Error) does not capture the outlier estimations, MMRE is biased towards the under estimation, and SA (Standardized Accuracy) is relative to the random guessing [43].

### 4.6 Statistical Analysis

To evaluate if the accuracy improvement of GPT2SP and other baselines is statistically significant with non-negligible effect size, we apply a non-parametric ScottKnott ESD test [47] instead of the original ScottKnott ESD approach.

The Non-Parametric ScottKnott ESD (NPSK) test is a multiple comparison approach that leverages a hierarchical clustering to partition the set of median values of techniques into statistically distinct groups with non-negligible difference. The NPSK test produces the ranking of treatment means while ensuring that (1) the magnitude of the difference for all of the treatments in each group is negligible;

and (2) the magnitude of the difference of treatments between groups is non-negligible.

The NPSK test is used to avoid any spurious results when the distributions do not meet the basic ANOVA assumptions of the original ScottKnott test (i.e., the normality of the distributions, the homogeneity of variances, and the minimum sample size). Therefore, the NPSK test is more appropriate to determine the statistical significance of the accuracy improvement of GPT2SP and other baselines as it does not require the assumptions of normal distributions, homogeneous distributions, and the minimum sample size. The NPSK test is made up of 2 steps:

1) *(Step 1) Find a partition that maximizes the median of each distribution between groups.* First, the distribution is sorted by the median value of the distributions. The Kruskal Chisq statistic is computed to identify a partition that maximizes the median values between groups. The Kruskal Chisq test is a non-parametric test, which does not require data normality and data heterogeneity assumptions.
2) *(Step 2) Split into two groups or merging into one group.* The magnitude of the difference for each pair for all of the treatment medians of the two groups is analyzed. If there is any one pair of treatment medians of two groups are non-negligible, the two groups are split, otherwise, the two groups are merged into one group. Then, the Cliff $|\delta|$ effect size is used to estimate the effect size of the difference between the two medians.

We used the implementation of the Non-Parametric ScottKnott ESD test from the ScottKnott ESD R package (Version 3.0).

## 5 Experimental Design and Results

*(RQ1) Does Our GPT2SP Outperform Deep-SE for Within-Project Scenarios?*

Approach. To address this RQ, we focus on within-project evaluation. Within-project evaluation is a scenario where models are trained and test on the dataset from one single project. To ensure that the within-project evaluation is realistic, the dataset is sorted in a chronological order for each project. To ensure a fair comparison with Deep-SE and avoid any temporal validation bias, the dataset of each project is split into three sets: training (60%), validation (20%), and testing (20%). The data split follows the temporal requirement where the training data came first followed by the validation data, and the testing data is the most recent data. This ensures that new issues in the testing set will not be used for training, and the past issues in the training set will not be used for testing. For each project, the GPT2SPmodels are trained on the training set, while the testing set is used to evaluate using the MAE measure. We apply the within-project evaluation scenario to each of the 16 different datasets. Then, for each dataset, we measure the MAE based on the model with the best hyper-parameter setting that achieves the lowest loss value. Note that the loss value is computed based on the validation data, while the MAE is measured based on the testing data.

Then, we compare our GPT2SP with nine existing baseline approaches, i.e., Deep-SE [7], LSTM+RF, LSTM+SVM, LSTM+ATLM, LSTM+LR, Doc2Vec+RF, BoW+RF, Mean, and Median. Similar to Choetkiertikul *et al.* [7], LSTM is used to generate a vector representation, then is fed into four ML techniques (i.e., Random Forest, Support Vector Machine, Automatically Transform Linear Models, Linear Regression). In addition, two additional feature representations are also used (i.e., Doc2Vec and Bag-of-Words) to generate a vector representation. Mean and Median Effort estimations are the mean or median story points of the past issues in the training set to estimate the story points of the target issue. [7], [45]. Finally, to quantify the magnitude of improvement for our GPT2SP with the nine baseline approaches, we compute a percentage improvement of the MAE using the following calculation: $\frac{(\text{MAE}_{\text{baseline}}-\text{MAE}_{\text{our}})\times 100\%}{\text{MAE}_{\text{baseline}}}$. Then, we apply a non-parametric ScottKnott ESD test to statistically rank the MAE distributions of our GPT2SP and the nine baseline approaches.

Results. Fig. 3 presents the percentage improvement of the MAE between our GPT2SP and the nine baseline approaches, while Fig. 4 presents the ranking of our GPT2SP and the nine

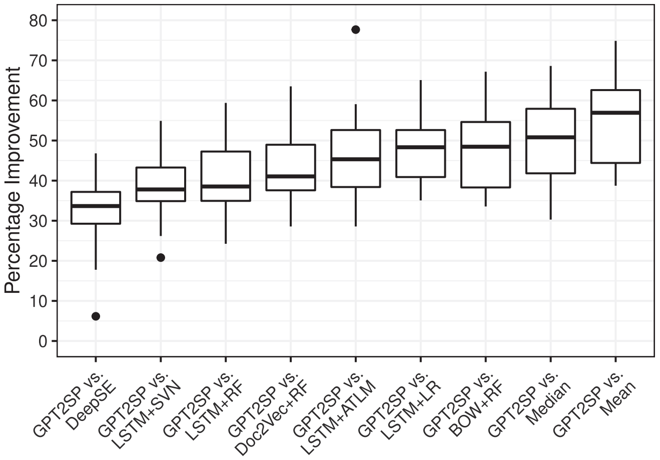


Fig. 3. (RQ1) The percentage improvement of the MAE between our GPT2SP and the nine baseline comparisons for within-project estimation scenarios.

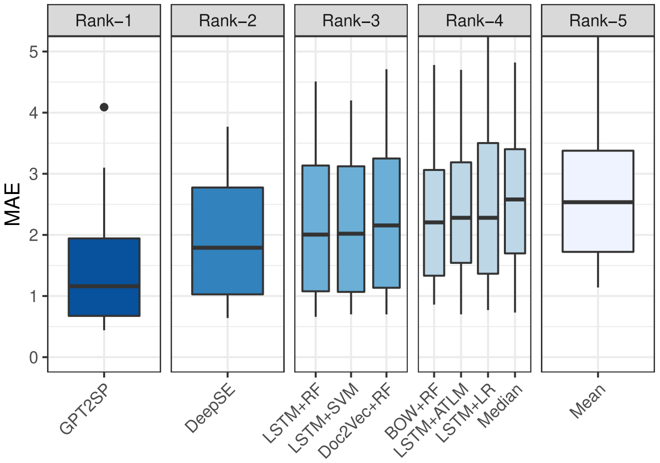


Fig. 4. (RQ1) The ranking of our GPT2SP and the nine baseline comparisons for within-project estimation scenarios. (↘) Lower Mean Absolute Error (MAE) = Better.

baseline approaches for within-project estimation scenarios. Each data point represents the MAE of each studied project.

*Our* GPT2SP *achieves a median MAE of 1.16, which is 34%-57% significantly more accurate than the existing nine baseline approaches.* Fig. 3 shows that our GPT2SPis 6%(Clover)-47% (Talend Data Quality) more accurate than Deep-SE and 38%-75% than the Mean baseline among the studied projects. In addition, Fig. 4 shows that our GPT2SP models are the most accurate for agile story point estimation. According to the non-parametric ScottKnott ESD ranking (see Fig. 4), our GPT2SP is the only approach that appeared in Rank-1, while Deep-SE appears in Rank-2 where the other 8 baseline approaches appear in Rank-3 to Rank-5, indicating that our GPT2SP models statistically outperform other existing baseline approaches with non-negligible difference for within-project evaluations. These results highlight the significant advancement of our GPT2SP approach over the state-of-the-art Agile story point estimation models.

*RQ2) Does Our GPT2SP Outperform Deep-SE for Cross-Project Scenarios?*

Approach. To address this RQ, we focus on cross-project evaluation. Cross-project evaluation is a scenario where models are trained from one project and test on another project. Similar to Choetkiertikul *et al.* [7], we focus on cross-repositories and within-repositories. For cross-repository evaluation, models are trained from a project in a repository, and tested on another project in the other repositories. For within-repository evaluation, models are trained from a project in a repository, and tested on another project within the same repository. For each project, the GPT2SPmodels are trained on the training set, while the testing set is used to evaluate using the MAE measure. To ensure a fair comparison, we use the same target project as Choetkiertikul *et al.* [7] for both cross-repository and within-repository evaluation scenarios Then, we compare our GPT2SP with Deep-SE [7] and Analogy-based estimation (ABE0) [19], [23], [24], [25]. The ABE0 estimates a story point of an issue in the target project based on the mean of story points of the $k$-nearest issues ($k$=3) from the source project. Similar to RQ1, we compute a percentage improvement of the MAE using the following calculation: $\frac{(\text{MAE}_{\text{baseline}}-\text{MAE}_{\text{our}})\times 100\%}{\text{MAE}_{\text{baseline}}}$. Then, we apply a non-parametric ScottKnott ESD test to statistically rank the MAE distributions of our GPT2SP and the two baseline approaches.

Results. Fig. 5 presents the ranking of our GPT2SP and the two baseline approaches for cross-project estimation scenarios. Each data point represents the MAE of each studied project.

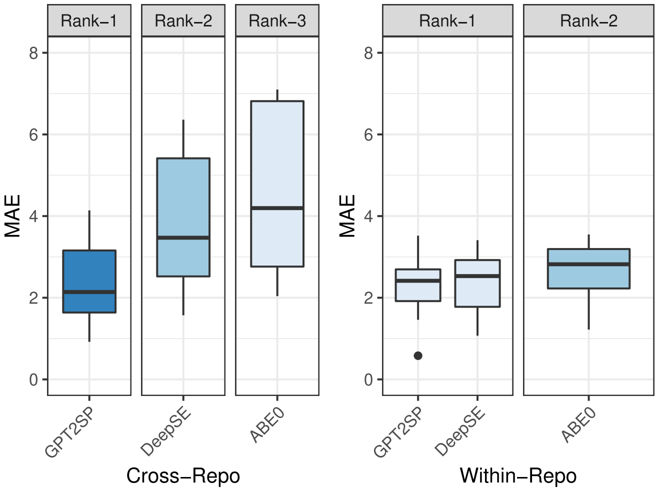


Fig. 5. (RQ2) The Mean Absolute Error (MAE) of our GPT2SP compared to Deep-SE and ABE0 for cross-project estimation scenarios.

*Regardless of the used training set from within-repository or cross-repository, our GPT2SP is still the most accurate for cross-project estimations when compared to Deep-SE and ABE0.* For cross-repository evaluations, our GPT2SPachieves a median MAE of 2.14, which is more accurate than than Deep-SE by 39% and ABE0 by 49%. We find that Deep-SE achieves a median MAE of 3.5, while ABE0 achieves a median MAE of 4.2. In particular, the improvement of our GPT2SP varies from 23% (Talend Data Quality →Aptana Studio) to 59% (Appcelerator Studio→Mule Studio). According to the non-parametric ScottKnott ESD ranking (see Fig. 5), our GPT2SP is the only approach that appears in Rank-1, while Deep-SE appears in Rank-2 and ABE0 appears in Rank-3, indicating that our GPT2SP statistically outperforms the other two baseline approaches with non-negligible difference for cross-repository evaluations.

The substantial improvement for cross-project estimation scenarios has to do with the use of GPT-2 pre-trained language models, highlighting the benefits of using GPT-2 language models to learn the distributed representations of words in a broader context than the Deep-SE project-specific pre-trained language models. In contrast, Deep-SE builds a pre-trained language model for each project to generate a vector representation of each word, limiting the understanding of the language models to the known words within the trained project and limiting the transferability of Deep-SE that learns from one project to other projects.

For within-repository evaluations, our GPT2SPis comparable to Deep-SE and ABE0. We find that GPT2SPachieves a median MAE of 2.4 (Rank-1), Deep-SE achieves a median MAE of 2.53 (Rank-1), while ABE0 achieves a median MAE of 2.82 (Rank-2). Despite the comparable performance, we find that GPT2SPoutperforms Deep-SE for 62.5% ($\frac{5}{8}$) of the within-repository evaluations. In particular, the improvement of our GPT2SP varies from 3% (Mule Studio→Mule) to 46% (Mesos→Usergrid).

*RQ3) What are the Contributions of the Components of GPT2SP?*

Approach. To address this RQ, we conduct an ablation study by examining the MAE of our GPT2SP when each component is varied. We consider the two major components of our GPT2SP, i.e., the BPE subword tokenization and the GPT-2 architecture. To better understand the contribution of the BPE subword tokenization, we keep the GPT-2 architecture, while varying the BPE subword tokenization to two alternative subword tokenization algorithms, i.e., $\text{WordPiece}_{\text{SP}}$ and $\text{SentencePiece}_{\text{SP}}$. To better understand the contribution of the GPT-2 Transformer, we keep the Word-Level tokenization (which was used in Deep-SE), while varying the architecture from LSTM+RHWN (used in Deep-SE) to GPT-2. In total, we compare the MAE of the five different models across the 16 datasets using the within-project scenario. Similar to RQ1, we measure the MAE based on the

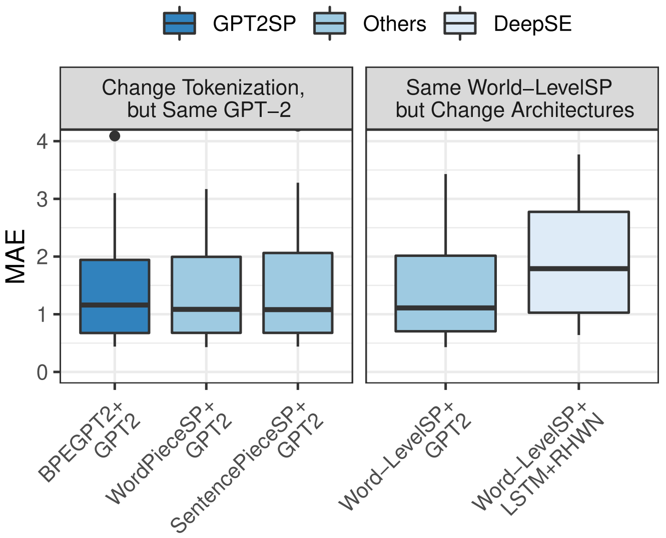


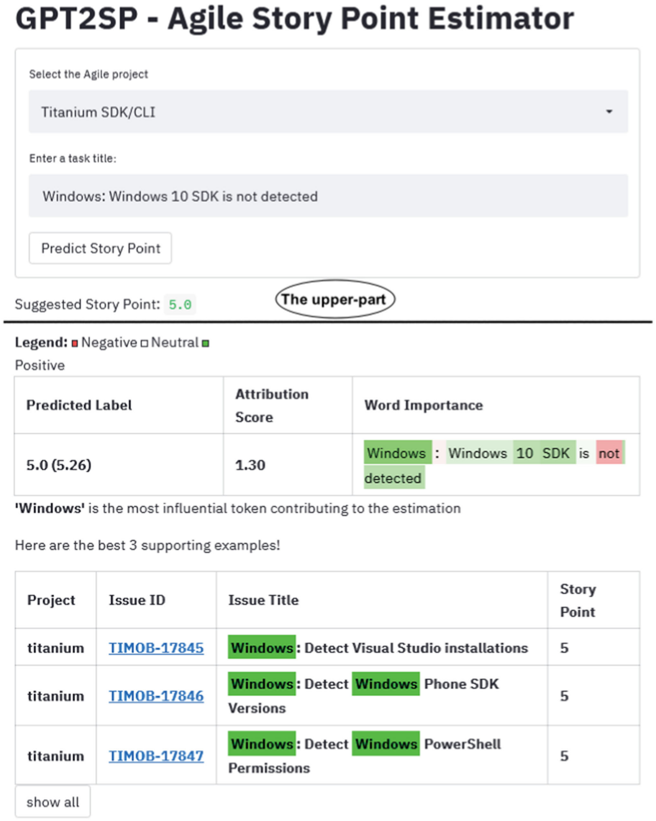
Fig. 6. (RQ3) The impact of our GPT2SP models when either of the tokenization or the architecture components is changed. This figure shows that the Transformer architecture substantially improves the MAE of Deep-SE (Word-LevelSP+LSTM+RHWN→Word-LevelSP+GPT2). On the other hands, the choice of subword tokenization algorithms have little impact on the MAE. Nevertheless, BPE is still the best subword toknization algorithm for our GPT2SP.

model with the best hyper-parameter setting that achieves the lowest loss value which is computed based on the validation data, while the MAE is measured based on the testing data.

*Results*. Fig. 6 presents the results of our GPT2SPand the other possible approaches.

*The Transformer architecture used in our GPT2SP substantially improves the MAE of Deep-SE by 6% to 47%*. When considering the different architectures with the same word-level tokenization (i.e., Word-LevelSP+GPT2 and Word-LevelSP+LSTM+ RHWN$_{\text{Deep-SE}}$), we observe that the MAE of Deep-SE is substantially improved by 6% to 47%, with a median percentage improvement of 34%. This substantial improvement highlights the benefits of the Transformer architecture which leverages the masked multi-head self-attention mechanism. Unlike the Deep-SE approach, their LSTM unit cannot capture all dependencies because the accumulated information in the short-term memory cell needs to be refreshed at each time step. Thus, some information may be discard overtime. However, the masked multi-head self-attention mechanism allows each word in a sequence to interact with each others equally with a masked mechanism to prevent the models to attend the subsequent positions, which better captures the dependencies between the words and provides richer semantic meanings.

On the other hand, when considering the different subword tokenization algorithms with the same GPT-2 architecture (i.e., BPE+GPT2$_{\text{GPT2SP}}$, WordPieceSP+GPT2, and SentencePieceSP+GPT2), we observe that different subword tokenization algorithms achieve little MAE differences. This finding indicates that, regardless of the subword tokenization algorithms, the GPT-2 architecture still outperforms the Deep-SE approach for all of the studied projects. We suspect that the little impact of the tokenization algorithms has to do with the different nature of the downstream tasks of the Transformer models. Recently, researchers raise concerns that different tokenization approaches have a large impact on the Transformer models for code generation tasks in SE (e.g., NMT-based automated program repairs [10], [16], [18]). Although BPE subword tokenization is employed in recent studies [10], [16], [18] to address the open/large vocabulary problem, our story point estimation is an estimation problem (Text→Number), which is different from the code generation tasks (Code→Code). Thus, the little impact of the tokenization approaches on the story point estimation models may suggest that open/large vocabulary problem is not a big concern for our story point estimation task. Nevertheless, BPE is still the best tokenization approach for our GPT2SP.



GPT2SP - Agile Story Point Estimator

Select the Agile project

Titanium SDK/CLI

Enter a task title:

Windows: Windows 10 SDK is not detected

Predict Story Point

Suggested Story Point: 5.0

The upper-part

Legend: 🟥 Negative □ Neutral 🟩 Positive

| Predicted Label | Attribution Score | Word Importance |
|---|---|---|
| 5.0 (5.26) | 1.30 | Windows : Windows 10 SDK is not detected |

'Windows' is the most influential token contributing to the estimation

Here are the best 3 supporting examples!

| Project | Issue ID | Issue Title | Story Point |
|---|---|---|---|
| titanium | TIMOB-17845 | Windows: Detect Visual Studio installations | 5 |
| titanium | TIMOB-17846 | Windows: Detect Windows Phone SDK Versions | 5 |
| titanium | TIMOB-17847 | Windows: Detect Windows PowerShell Permissions | 5 |

show all

Fig. 7. An example screenshot of our web-based GPT2SP tool (URL: https://share.streamlit.io/awsm-research/gpt2sp_webapp/main/app.py).

## 6 A Proof-of-Concept Explainable AI-Based Story Point Estimation With Explanations and Supporting Examples

In this section, we present a proof-of-concept of our explainable AI-based story point estimation (called GPT2SP) as a web-based tool with explanations and supporting examples. The main purpose of the proof-of-concept is used to conduct a survey study in order to investigate the challenge of story point estimation tasks and motivate the need of supporting explanations for AI-based story point estimation.

### 6.1 A GPT2SP Tool

To help practitioners better understand the estimation and promote the adoption of our GPT2SP, we develop a proof-of-concept web-based story point estimation tool (see Fig. 7). Taking an issue as input, our tool is designed to serve three purposes:

1) *Estimate* the story point;
2) *Highlight* the most important word that contributed to the story point estimation of the given issue; and
3) *Provide* supporting examples from the training set of the same project. Our *supporting examples* refer to example issues in the training set of the same project that contain the most important word with the same story point.

In particular, we leverage two concepts of Explainable AI for our GPT2SP tool, i.e., feature-based explanations and example-based explanations. Feature-based explanations aim to help practitioners better understand what are the most important word that contributed to the story point estimation of the given issue. On the other hand, example-based explanations extend the concept of case-based reasoning [2] by searching for the best supporting examples that have the same word and the same story point from the same project.

*An Example Usage of our GPT2SP tool*. To illustrate an example usage of our GPT2SP tool, we select an issue (TIMOB-20252)[1] from the Titanium project. The title of TIMOB-20252 is *"Windows: Windows 10 SDK is not detected"*. Once we input this title into our GPT2SP tool, the model estimates a story point of 5.0. We find that the story point estimation for this issue is correct based on the actual ground-truth.

Then, our GPT2SP tool will provide 2 key explanations: (1) what are the most important word that contributed to the story point estimation of the given issue; and (2) what are the best supporting examples from the past estimates. Our GPT2SP tool shows that "Windows" is the most important word that contributes to the story point estimation of this issue. In addition, our GPT2SP tool also shows the top-3 supporting examples according to the most important word (i.e., TIMOB-17845,[2] TIMOB-17846,[3] TIMOB-17847[4]). We find that these three supporting examples have the word "Windows" and the same story points, indicating that *issues with similar story points often share similar keywords*. In other words, this highlights that our GPT2SP may help Agile teams achieve the consistency of story point estimation based on historical data.

*Implementation Details*. We develop the web-based GPT2SP using the Streamlit framework.[5] First, we publish the GPT2SP models on the HuggingFace's Model Hub [56]. These models will be imported according to the selected project. Then, our GPT2SP tool will estimate a story point according a given input. Then, we extract the word attribution scores from the attention weights of the masked multi-head self-attention mechanism. The word attribution score is calculated based on Integrated Gradient, which is an axiomatic model interpretability algorithm [46]. We use the implementation of the Integrated Gradient approach provided by the transformer-interpret Python library.[6] The word attribution score indicates the contribution of each word in an issue to the final story point estimate.

Fig. 7 shows an example of highlighted words according to their attribution scores. The word attribution score varies from -1 to 1, where the positive ranges indicate the positive contribution (highlighted in Green), while the negative ranges indicate the negative contribution (highlighted in Red). Then, we search for supporting examples from the past estimates (i.e., issues in the training set of that project with the same word and the same story point). In this example, we find three best supporting examples from the past estimates (i.e., TIMOB-17845, TIMOB-17846, TIMOB-17847).

### 6.2 A Survey Study With Agile Practitioners

The purpose of our survey is to understand the practitioners' perceptions of (1) the challenge of the story point estimation task and (2) the AI-based story point estimation with/without explanations. Similar to Kitchenham and Pfleeger [22], we conducted our study according to the following steps: (1) design and develop a survey, (2) recruit and select participants, and (3) verify data and analyze data. We explained the detail of each step below.

*(Step 1) Design and Develop a Survey.* We designed our survey as a cross-sectional study where participants provided their responses at one fixed point in time. The survey consists of 10 closed-ended questions and 3 open-ended questions. For closed-ended questions, we used multiple-choices question and Likert scale from 1 to 5. Our survey consists of three parts: preliminary question; AI-based agile story point estimations; and AI-based agile story point estimations with explanations.

*Part I: Demographics and Challenge.* The survey starts with a question ("Do you have experience working in the context of Agile Software Development?") to ensure that our survey results obtained from the right target participants. Then, the survey is followed by demographics questions related to roles, levels of experience, whether the participant estimate story points, and the degree of challenge of the story point estimation task followed by an open question for the rationale.

*Part II: Story point estimation without explanations.* The next part focuses on practitioners' perceptions on AI-based story point estimations without explanations. Specifically, we presented a usage scenario and an example visualization of an AI-based story point estimation without explanations using the upper-part visualization of Fig. 7 Then, we asked two questions, i.e., ("How do you perceive the usefulness of the AI-based Agile Story Point estimations?") and ("Do you trust the AI-based Agile Story Point estimations?") followed by an open question for the rationale.

*Part III: Story point estimation with explanations.* Finally, we repeated the same questions as Part II, but with explanations using the complete visualization of Fig. 7.

We used Google Form to conduct our survey in an online setting. Each participant is provided with an explanatory statement on the landing page that describes the purpose of the study, why the participant is chosen for this study, possible benefits and risks, and confidentiality. The survey takes approximately 10 minutes to complete and is completely anonymous. Our survey has been rigorously reviewed and approved by the Monash University Human Research Ethics Committee (MUHREC ID: 31180).

[1] https://jira.appcelerator.org/browse/TIMOB-20252
[2] https://jira.appcelerator.org/browse/TIMOB-17845
[3] https://jira.appcelerator.org/browse/TIMOB-17846
[4] https://jira.appcelerator.org/browse/TIMOB-17847
[5] https://streamlit.io/
[6] https://github.com/cdpierse/transformers-interpret

*(Step 2) Recruit and Select Participants.* We recruited the practitioners that have software engineering related experience through LinkedIn and Facebook platforms. We sent a survey invitation to the target groups via the direct message. To ensure that our survey is not biased, we selected participants from various large companies. Finally, we obtained a total of 33 responses over a two-weeks period of recruitment.

*(Step 3) Verify Data and Analyze Data.* To ensure that our survey results are derived from the right target participants, we exclude 17 responses that do not have experience working in the context of Agile Software Development and do not perform story point estimation. To verify the completeness of the response in our survey (i.e., whether all questions were appropriately answered), we manually review all of the open-ended questions. Finally, we obtain a set of 16 responses. We present the results of closed-ended responses in a Likert scale with stacked bar plots. We manually analyze the responses of the open-ended questions to better understand the in-depth insights. Respondent demographics are provided in Appendix, available in the online supplemental material.

### 6.3 Survey Results

Figure 8 presents the results of our survey study with 16 Agile practitioners.

*(Q1) How do practitioners perceive the challenge of the story point estimation task?*

*Findings. 38% of the respondents perceived that the story point estimation task is challenging to extremely challenging* due to various reasons:

- Heterogeneity of team members (R1: *Because we need to estimate story point to match all people in our team.*)
- Subjective estimation (R3: *It is quite subjective, depending on team members' experience.*)
- Task familiarity (R3: *Some tasks outside of your area are difficult to estimate how long they should take, comparing with the tasks you're familiar with.*, R8: *It is difficult to know the exact time for the story point, especially for the task that we haven't worked on it before.*)
- Project size and group size (R10: *The larger the project and group, the more challenging the estimation.*)
- Problem description is simplified inappropriately (R13: *Question description (the problem to be solved) is always too simplistic.*)

*(Q2) Would AI-based story point estimation with explanations be more useful and trustworthy than without explanations?*

*Findings. AI-based story point estimation with explanations (62% of respondents) is perceived more useful than without explanations (50% of respondents).* Similarly, AI-based story point estimation with explanations (50% of the respondents) is perceived more trustworthy than without explanations (26% of respondents). AI-based story point estimation with explanations is perceived as more useful and trustworthy due to the following reasons:

- The existence of supporting explanations (R1: *More trust after seeing supporting information but not fully trust right now.*, R12: *It looks more sensible with supporting explanations and also helps the development to make his/her own judgement.*)
- The knowledge of historical training data (R3: *It based on previous estimation but some outliers might take more effort than usual so tagging might not be able to help.*)
- The supporting explanations make the estimation more reasonable (R13: *It seems more reasonable.*)

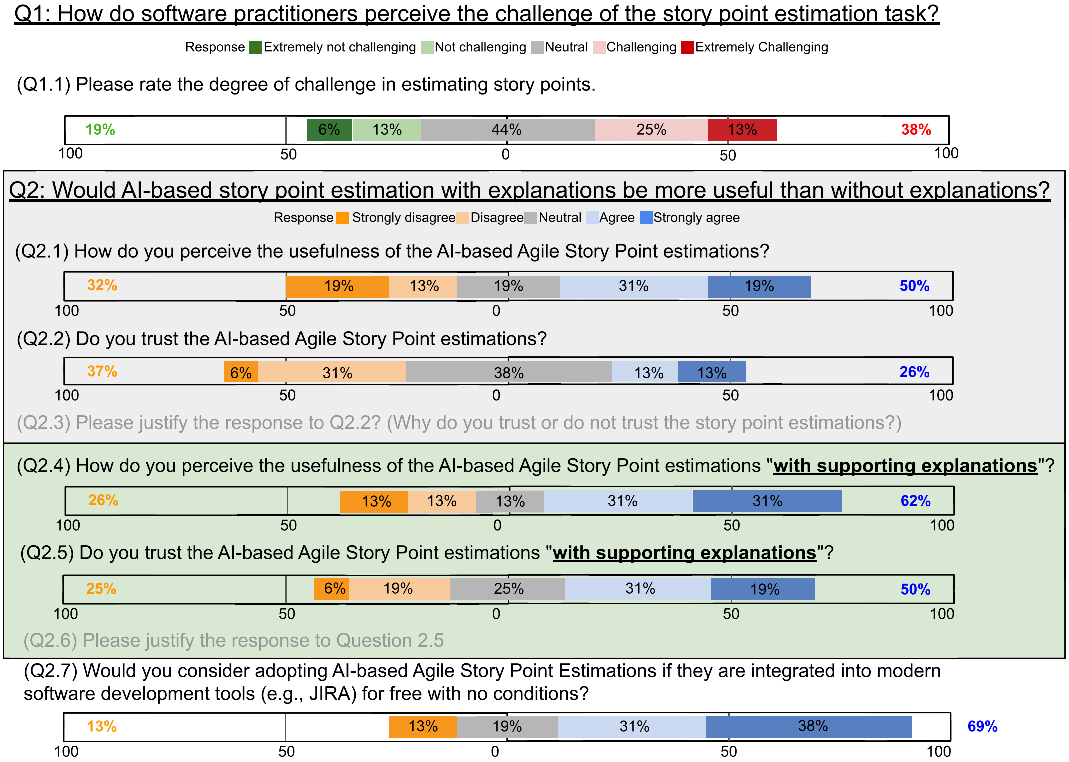


Fig. 8. (Q1/Q2) A summary of the survey questions and the results obtained from 16 participants.

- Feel more comfortable to leverage the estimation with supporting explanations (R8: *By knowing where the suggestion is come from and is the result based on the related task or not. It make me feel more comfortable to decide whether to use the predicted story point for that task or not.*)

Summary. Our survey study with 16 Agile practitioners found that the story point estimation task is perceived as an extremely challenging task. In addition, our AI-based story point estimation with explanations is perceived as more useful and trustworthy than without explanations. Finally, we found that 69% of the respondents consider adopting AI-based Agile Story Point Estimations if they are integrated into modern software development tools (e.g., JIRA), highlighting the practical need of our Explainable AI-based story point estimation approach.

## 7 Discussion

Below, we discuss how our work will be beneficial to practitioners and researchers.

Benefits to Practitioners. The primary goal of our GPT2SP is to augment Agile practitioners in estimating story points prior to the sprint plannings according to the past estimates to ensure the consistency among the team members—*not to achieve a fully-automated Agile process*. Instead, our GPT2SP aims to serve as a decision-support system to augment human-decision makings for the story point estimation practices. Like any socio-technical projects, the Human-AI interaction is still required, meaning that the team meeting is needed during the backlog refinement, but our GPT2SP will provide story point estimations that align with past estimates.

Benefits to Researchers. To the best of our knowledge, this paper is the first to leverage a Transformer architecture for Agile story point estimations. Our RQ1 and RQ2 show that our GPT2SP outperforms ten existing baseline approaches, while our RQ3 shows that the Transformer architecture used by our GPT2SP yields greatly improves the accuracy of the story point estimation when compared to the state-of-the-art Deep-SE. Thus, the customized Transformer architecture for our regression task may be applicable to other SE tasks.

In addition, this paper is the first to leverage the masked multi-head self-attention mechanism inside the Transformer to highlight what words contribute the most to the estimation, making our GPT2SP approach more interpretable than Deep-SE. In addition, we also develop a proof-of-concept tool to provide the best supporting examples from past estimates. The supporting examples provided by our GPT2SP tool could be an example model inspection analysis for other AI4SE tasks, highlighting a promising research direction towards Explainable AI for Software Engineering (XAI4SE), which is still challenging and rarely explored.

## 8 Related Work

In this section, we discuss related work and the difference of our work with respect to the existing literature.

### 8.1 Effort Estimation

Effort estimation is proposed to estimate the amount of effort (e.g., person hours) that is required to complete the development of software products. Effort estimation is essential to help teams gain a better understanding of how much time, effort, and money it will take to deliver software products that are of value to their organizations. For example, Boehm *et al.* propose a COnstruction COst MOdel (COCOMO) [5], which is a regression-based approach that leverages the predefined factors and their relationships with the estimated effort. The COCOMO model is built using a series of past projects, therefore, the performance is limited when facing a new project that has not appeared in past data. Commeyne *et al.* [8] propose an approach which relies on COSMIC Function Points (CFPs) and to estimate the effort for completing an agile project.

Difference. Different from prior studies in effort estimation which is proposed for traditional software development like waterfall, this paper focuses on Agile story point, which is another proxy of effort measurement in the context of Agile software development.

### 8.2 ML-Based Story Point Estimation

ML-based story point estimation is proposed to help teams to accurately estimate Agile story points for a given work item by leveraging Machine Learning (ML) techniques. In particular, Story Point Estimation is formulated as a regression task to estimate a story point given an input feature vector. Previously, researchers propose to use various handcrafted features (e.g., working hours and developer skills [11]) for estimating story points with various ML-based models (e.g., Bayesian network [11], [14], support vector machine [36]). However, the traditional data collection of such handcrafted features is still manual and time-consuming that could potentially hinder the adoption in practice.

To address the challenge of manual feature engineering process, Porru *et al.* [36] use the document frequency (TFIDF [41]) to extract features from text data with a support vector machine (SVM) classifier to predict the story point. As this approach is able to generate feature representation without manual extraction, such approach needs a document cleaning process (e.g., eliminating special characters) and only focuses on the document frequency without extracting the semantic and syntactic between words, that could lead to meaningless feature representation.

Difference. Different from prior studies in ML-based story point estimation, our GPT2SP leverages a Transformer architecture, which is a deep learning approach, to automatically learn the semantic of textual features in the issue reports.

### 8.3 DL-Based Story Point Estimation

Various deep learning-based story point estimation approaches have been proposed. For example, Panda *et al.* [29] compare different types of Neural Networks (e.g., General Regression Neural Network and Polynomial Neural Network) for story point estimation. Praynlin *et al.* [37] leverage Recurrent Neural Network, ELMAN Neural Network, and explore the effectiveness of using backpropagation algorithm to train a Neural Network for story point estimation. Marapelli *et al.* [27] propose an approach based on RNN and Convolutional Neural Networks (CNN). Choetkiertikul *et al.* [7] propose an approach based on

Long Short-Term Memory (LSTM) with a Recurrent Highway Networks (RHWN).

Difference. Different from prior studies that focus more on exploring traditional NN and RNN-based models for story point estimation, our paper is the first to leverage a Transformer architecture and the pre-trained language model to address various limitations of Deep-SE [7]. Our results show significant performance improvement with non-negligible effect size. In addition, this paper is the first to leverage the attention mechanism to provide supporting explanations to practitioners, as our survey results show that with supporting explanations are perceived more useful and more trustworthy than without supporting explanations.

### 8.4 Explainable AI for SE

The explainability of AI models in SE recently becomes a critical concern, since practitioners often do not trust the predictions [49], which may hinder the adoption of AI models for SE in practice. Recently, Explainable AI has been actively investigated in the domain of defect prediction [48], [49]. For example, recent works have shown some successful case studies to make defect prediction models more practical [15], [33], [34], [55], explainable [17], [20], and actionable [31], [35], [40]. However, these studies only focus on explaining the traditional ML approaches.

Difference. To the best of our knowledge, this paper is among the first to leverage the attention mechanism of the GPT architecture for explaining deep learning-based story point estimation models. Unlike prior studies, our GPT2SP is a deep learning approach that is designed to be interpretable by using the self-attention mechanism.

### 8.5 Attention Mechanism as a Model Interpretability Technique

Deep learning is known to be complex and hardly interpretable. Thus, various model-agnostic techniques from the Explainable AI literature are proposed to explain the predictions of deep learning or machine learning models, which have been recently adopted in software engineering. For example, recent studies [17], [20], [31], [33], [35], [40], [55] leverage a LIME model-agnostic technique [42] for explaining the predictions of defect prediction models. However, such model-agnostic techniques are considered as an *extrinsic* model-agnostic technique (i.e., a model-agnostic is applied to explain a black-box DL/ML model after the model is trained), not an *intrinsic* model-agnostic technique (i.e., a DL/ML model that is interpretable by itself so extrinsic model-agnostic techniques are not needed to apply afterward).

Difference. Different from prior studies, this paper is among the first attempt to leverage the attention mechanism to explain the prediction of Transformer-based agile story point estimation models, highlighting the substantial benefits of the attention mechanism for story point estimation.

## 9 Threats to Validity

In this section, we discuss threats to the validity.

*Threats to construct validity* relate to the quality of story point datasets. Prior studies raised concerns that the quality of datasets may have an impact on the accuracy of machine learning models, especially, for defect prediction [13], [32], [50]. In particular, prediction models may be inaccurate if the ground-truth labels are noisy (e.g., a NASA defect dataset). Such noisy labels (i.e., a file is labeled as defective or not) in defect datasets have to do with the use of heuristics to generate ground-truths data. On the other hand, story point datasets prepared by Choetkiertikul *et al.* [7] are labelled based on actual information provided in the JIRA issue tracking system, *which is not from any heuristics*. Based on our random samples, we do not observe any bias in the ground-truths. Thus, the quality of story point datasets may not pose a great risk to our experimental results.

*Threats to internal validity* relate to the hyperparameter settings of GPT2SP. Our GPT2SP consists of various hyperparameter settings (i.e., number of hidden layers, number of attention heads, and learning rate). Prior studies raise concerns that different hyperparameter settings may have an impact on the evaluation results, especially, for defect prediction models [47], [51]. However, finding an optimal hyperparameter setting can be very expensive given a large search space of the Transformer architecture. Instead, the goal of our work is not to find the best hyperparameter setting, but to fairly compare the accuracy of our approach with the existing baseline approaches. Thus, the accuracy reported in the paper is served as a lower bound of our approach, which can be even further improve through hyperparameter optimization. To mitigate this threat, we report the hyperparameter settings in the replication package to aid future replication studies.

*Threats to the external validity* relate to the generalizability of the accuracy of our GPT2SP approach. The evaluation results of our GPT2SP are based on the datasets of 23,313 issues provided by Choetkiertikul *et al.* [7]. Thus, the results may not be generalized to other datasets and other open-source or proprietary software projects. Therefore, other datasets can be explored in future work.

The results of our survey study are limited to 16 Agile practitioners. Thus, our findings may not be generalized to other practitioners and companies with different processes, settings, and cultures. On the other hand, our survey study with 16 Agile practitioners has shown that our Explainable AI-based Agile story point tool is useful to some extent. Thus, a user study with real-world settings is recommended in future work.

## 10 Conclusion

In this paper, we propose GPT2SP, a Transformer-based Agile Story Point Estimation approach to address the three limitations of Deep-SE. Through an extensive evaluation of our approach on 23,313 issues that span across 16 open-source software projects with 10 existing baseline approaches for within- and cross-project scenarios, our results show that our GPT2SP approach (1) is 34%-57% more accurate than existing baseline approaches for within-project estimations; (2) is 39%-49% more accurate than existing baseline approaches for cross-project estimations. The ablation study also shows that the GPT-2 architecture used in our approach substantially improves Deep-SE by 6%-47%, highlighting the significant advancement of the AI for Agile story point estimation. Finally, our survey study with 16 Agile practitioners shows that AI-based story point estimation with explanations is perceived as more useful and trustworthy than without explanations. In addition, we find that 69% of the respondents consider adopting AI-based Agile Story Point Estimations if they are integrated into modern software development tools (e.g., JIRA), highlighting the practical need of our Explainable AI-based story point estimation approach.

## Acknowledgments

We thank Choetkiertikul *et al.* [7] for providing the datasets and the replication package. With similar effort, we also publish a replication package that contains datasets, implementation scripts, raw predictions, the GPT2SP tool, and models available in HuggingFace to promote open science and the adoption of our GPT2SP tool.

## References

[1] Story points and estimation, 2020. [Online]. Available: https://www.atlassian.com/agile/project-management/estimation

[2] A. Aamodt and E. Plaza, “Case-based reasoning: Foundational issues, methodological variations, and system approaches,” *AI Commun.*, vol. 7, no. 1, pp. 39–59, 1994.

[3] P. Abrahamsson, O. Salo, J. Ronkainen, and J. Warsta, “Agile software development methods: Review and analysis,” 2017, *arXiv: 1709.08439*.

[4] J. Bergstra, D. Yamins, and D. Cox, “Making a science of model search: Hyperparameter optimization in hundreds of dimensions for vision architectures,” in *Proc. Int. Conf. Mach. Learn.*, 2013, pp. 115–123.

[5] B. W. Boehm *et al.*, *Software Cost Estimation with Cocomo II*. Englewood Cliffs, NJ, USA: Prentice Hall Press, 2009.

[6] R. Bommasani *et al.*, “On the opportunities and risks of foundation models,” 2021, *arXiv:2108.07258*.

[7] M. Choetkiertikul, H. K. Dam, T. Tran, T. Pham, A. Ghose, and T. Menzies, “A deep learning model for estimating story points,” *IEEE Trans. Softw. Eng.*, vol. 45, no. 7, pp. 637–656, Jul. 2019.

[8] C. Commeyne, A. Abran, and R. Djouab, “Effort estimation with story points and cosmic function points-an industry case study,” *Softw. Meas. News*, vol. 21, no. 1, pp. 25–36, 2016.

[9] H. K. Dam, T. Tran, and A. Ghose, “Explainable software analytics,” in *Proc. Int. Conf. Softw. Eng.: New Ideas Emerg. Results*, 2018, pp. 53–56.

[10] Y. Ding, B. Ray, P. Devanbu, and V. J. Hellendoorn, “Patching as translation: The data and the metaphor,” in *Proc. 35th IEEE/ACM Int. Conf. Automated Softw. Eng.*, 2020, pp. 275–286.

[11] S. Dragicevic, S. Celar, and M. Turic, “Bayesian network model for task effort estimation in Agile software development,” *J. Syst. Softw.*, vol. 127, pp. 109–119, 2017.

[12] P. Gage, “A new algorithm for data compression,” *C Users J.*, vol. 12, no. 2, pp. 23–38, 1994.

[13] B. Ghotra, S. McIntosh, and A. E. Hassan, “Revisiting the impact of classification techniques on the performance of defect prediction models,” in *Proc. Int. Conf. Softw. Eng.*, 2015, pp. 789–800.

[14] P. Hearty, N. Fenton, D. Marquez, and M. Neil, “Predicting project velocity in XP using a learning dynamic Bayesian network model,” *IEEE Trans. Softw. Eng.*, vol. 35, no. 1, pp. 124–137, Jan./Feb. 2009.

[15] Y. Hong, C. K. Tantithamthavorn, and P. P. Thongtanunam, “Where should I look at? Recommending lines that reviewers should pay attention to,” in *Proc. IEEE Int. Conf. Softw. Anal. Evol. Reeng.*, 2022.

[16] N. Jiang, T. Lutellier, and L. Tan, “CURE: Code-aware neural machine translation for automatic program repair,” in *Proc. Int. Conf. Softw. Eng.*, 2021, pp. 1161–1173.

[17] J. Jiarpakdee, C. K. Tantithamthavorn, H. K. Dam, and J. Grundy, “An empirical study of model-agnostic techniques for defect prediction models,” *IEEE Trans. Softw. Eng.*, vol. 48, no. 1, pp. 166–185, Jan. 2022.

[18] R.-M. Karampatsis, H. Babii, R. Robbes, C. Sutton, and A. Janes, “Big code != Big vocabulary: Open-vocabulary models for source code,” in *Proc. IEEE/ACM 42nd Int. Conf. Softw. Eng.*, 2020, pp. 1073–1085.

[19] J. W. Keung, B. A. Kitchenham, and D. R. Jeffery, “Analogy-X: Providing statistical inference to analogy-based software cost estimation,” *IEEE Trans. Softw. Eng.*, vol. 34, no. 4, pp. 471–484, Jul./Aug. 2008.

[20] C. Khanan *et al.*, “JITBot: An explainable just-in-time defect prediction bot,” in *Proc. 35th IEEE/ACM Int. Conf. Automated Softw. Eng.*, 2020, pp. 1336–1339.

[21] D. P. Kingma and J. Ba, “Adam: A method for stochastic optimization,” 2014, *arXiv:1412.6980*.

[22] B. A. Kitchenham and S. L. Pfleeger, “Personal opinion surveys,” in *Guide to Advanced Empirical Software Engineering*, Berlin, Germany: Springer, 2008, pp. 63–92.

[23] E. Kocaguneli, T. Menzies, A. Bener, and J. W. Keung, “Exploiting the essential assumptions of analogy-based effort estimation,” *IEEE Trans. Softw. Eng.*, vol. 38, no. 2, pp. 425–438, Mar./Apr. 2012.

[24] E. Kocaguneli, T. Menzies, and E. Mendes, “Transfer learning in effort estimation,” *Empir. Softw. Eng.*, vol. 20, no. 3, pp. 813–843, 2015.

[25] Y.-F. Li, M. Xie, and T. N. Goh, “A study of project selection and feature weighting for analogy based software cost estimation,” *J. Syst. Softw.*, vol. 82, no. 2, pp. 241–252, 2009.

[26] I. Loshchilov and F. Hutter, “Decoupled weight decay regularization,” 2017, *arXiv:1711.05101*.

[27] B. Marapelli, A. Carie, and S. M. N. Islam, “RNN-CNN model: A bi-directional long short-term memory deep learning network for story point estimation,” in *Proc. 5th Int. Conf. Innov. Technol. Intell. Syst. Ind. Appl.*, 2020, pp. 1–7.

[28] A. Mastropaolo *et al.*, “Studying the usage of text-to-text transfer transformer to support code-related tasks,” in *Proc. IEEE/ACM 43rd Int. Conf. Softw. Eng.*, 2021, pp. 336–347.

[29] A. Panda, S. M. Satapathy, and S. K. Rath, “Empirical validation of neural network models for Agile software effort estimation based on story points,” *Procedia Comput. Sci.*, vol. 57, pp. 772–781, 2015.

[30] A. Paszke *et al.*, “PyTorch: An imperative style, high-performance deep learning library,” in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2019, pp. 8026–8037.

[31] K. Peng and T. Menzies, “Defect reduction planning (using TimeLIME),” *IEEE Trans. Softw. Eng.*, to be published, doi: 10.1109/TSE.2021.3062968.

[32] J. Petrić, D. Bowes, T. Hall, B. Christianson, and N. Baddoo, “The Jinx on the NASA software defect data sets,” in *Proc. Int. Conf. Eval. Assessment Softw. Eng.*, 2016, pp. 13–17.

[33] C. Pornprasit and C. K. Tantithamthavorn, “JITLine: A simpler, better, faster, finer-grained just-in-time defect prediction,” in *Proc. IEEE/ACM 18th Int. Conf. Mining Softw. Repositories*, 2021, pp. 369–379.

[34] C. Pornprasit and C. Tantithamthavorn, “DeepLineDP: Towards a deep learning approach for line-level defect prediction,” *IEEE Trans. Softw. Eng.*, to be published, doi: 10.1109/TSE.2022.3144348.

[35] C. Pornprasit, C. Tantithamthavorn, J. Jiarpakdee, M. Fu, and P. Thongtanunam, “PyExplainer: Explaining the predictions of just-in-time defect models,” in *Proc. 36th IEEE/ACM Int. Conf. Automated Softw. Eng.*, 2021, pp. 407–418.

[36] S. Porru, A. Murgia, S. Demeyer, M. Marchesi, and R. Tonelli, “Estimating story points from issue reports,” in *Proc. 12th Int. Conf. Predictive Models Data Anal. Softw. Eng.*, 2016, pp. 1–10.

[37] E. Praynlin and P. Latha, “Performance analysis of software effort estimation models using neural networks,” *Int. J. Inf. Technol. Comput. Sci.*, vol. 5, no. 9, pp. 101–107, 2013.

[38] A. Radford, K. Narasimhan, T. Salimans, and I. Sutskever, “Improving language understanding by generative pre-training,” 2018.

[39] A. Radford *et al.*, “Language models are unsupervised multitask learners,” *OpenAI Blog*, vol. 1, no. 8, 2019, Art. no. 9.

[40] D. Rajapaksha, C. Tantithamthavorn, J. Jiarpakdee, C. Bergmeir, J. Grundy, and W. Buntine, “SQAPlanner: Generating data-informed software quality improvement plans,” *IEEE Trans. Softw. Eng.*, to be published, doi: 10.1109/TSE.2021.3070559.

[41] J. Ramos *et al.*, “Using TF-IDF to determine word relevance in document queries,” in *Proc. 1st Instructional Conf. Mach. Learn.*, 2003, pp. 29–48.

[42] M. T. Ribeiro, S. Singh, and C. Guestrin, “Why should I trust you?: Explaining the predictions of any classifier,” in *Proc. Int. Conf. Knowl. Discov. Data Mining*, 2016, pp. 1135–1144.

[43] F. Sarro, A. Petrozziello, and M. Harman, “Multi-objective software effort estimation,” in *Proc. IEEE/ACM 38th Int. Conf. Softw. Eng.*, 2016, pp. 619–630.

[44] R. Sennrich, B. Haddow, and A. Birch, “Neural machine translation of rare words with subword units,” 2015, *arXiv:1508.07909*.

[45] M. Shepperd and S. MacDonell , "Evaluating prediction systems in software project estimation," *Inf. Softw. Technol.*, vol. 54, no. 8, pp. 820–827, 2012.

[46] M. Sundararajan, A. Taly, and Q. Yan, "Axiomatic attribution for deep networks," in *Proc. Int. Conf. Mach. Learn.*, 2017, pp. 3319–3328.

[47] C. Tantithamthavorn, A. E. Hassan, and K. Matsumoto, "The impact of class rebalancing techniques on the performance and interpretation of defect prediction models," *IEEE Trans. Softw. Eng.*, vol. 46, no. 11, pp. 1200–1219, Nov. 2020.

[48] C. Tantithamthavorn, J. Jiarpakdee, and J. Grundy, "Explainable AI for software engineering," 2020, *arXiv: 2012.01614*.

[49] C. Tantithamthavorn, J. Jiarpakdee, and J. Grundy, "Actionable analytics: Stop telling me what it is; Please tell me what to do," *IEEE Softw.*, vol. 38, no. 4, pp. 115–120, Jul./Aug. 2021.

[50] C. Tantithamthavorn, S. McIntosh, A. E. Hassan, A. Ihara, and K. Matsumoto, "The impact of mislabelling on the performance and interpretation of defect prediction models," in *Proc. Int. Conf. Softw. Eng.*, 2015, pp. 812–823.

[51] C. Tantithamthavorn, S. McIntosh, A. E. Hassan, and K. Matsumoto, "Automated parameter optimization of classification techniques for defect prediction models," in *Proc. Int. Conf. Softw. Eng.*, 2016, pp. 321–332.

[52] M. Usman, E. Mendes, and J. Börstler, "Effort estimation in agile software development: A survey on the state of the practice," in *Proc. 19th Int. Conf. Eval. Assessment Softw. Eng.*, 2015, pp. 1–10.

[53] M. Usman, E. Mendes, F. Weidt, and R. Britto, "Effort estimation in agile software development: A systematic literature review," in *Proc. 10th Int. Conf. Predictive Models Softw. Eng.*, 2014, pp. 82–91.

[54] A. Vaswani *et al.*, "Attention is all you need," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2017, pp. 5998–6008.

[55] S. Wattanakriengkrai, P. Thongtanunam, C. Tantithamthavorn, H. Hata, and K. Matsumoto, "Predicting defective lines using a model-agnostic technique," *IEEE Trans. Softw. Eng.*, to be published, doi: 10.1109/TSE.2020.3023177.

[56] T. Wolf *et al.*, "HuggingFace's transformers: State-of-the-art natural language processing," 2019, *arXiv:1910.03771*.

**Michael Fu** (Student Member, IEEE) is currently working toward the PhD degree at Monash University, Melbourne, Australia. His research interests include AI-based application on software engineering problems.

**Chakkrit Tantithamthavorn** (Member, IEEE) is a senior research fellow with the Faculty of Information Technology, Monash University, Australia. He is recognized as the most impactful early-career SE researcher based on a bibliometric assessment of software engineering (2013–2020), and received numerous prestigious awards including an ACM SIGSOFT Distinguished Paper Award at ASE'21, and a 2020 ARC's Discovery Early Career Researcher Award (DECRA).