Evaluation of Context-Aware Language Models and Experts for Effort Estimation of Software Maintenance Issues

Mohammed Alhamed
School of Computing Science
University of Glasgow
Glassgow, UK
m.alhamed.1@research.gla.c.uk

Tim Storer
School of Computing Science
University of Glasgow
Glassgow, UK
timothy.storer@glasgow.ac.uk

***Abstract*—Reflecting upon recent advances in Natural Language Processing (NLP), this paper evaluates the effectiveness of context-aware NLP models for predicting software task effort estimates. Term Frequency–Inverse Document Frequency (TF-IDF) and Bidirectional Encoder Representations from Transformers (BERT) were used as feature extraction methods; Random forest and BERT feed-forward linear neural networks were used as classifiers. Using three datasets drawn from open-source projects and one from a commercial project, the paper evaluates the models and compares the best performing model with expert estimates from both kinds of datasets. The results suggest that BERT as a feature extraction and classifier shows slightly better performance than other combinations, but that there is no significant difference between the presented methods. On the other hand, the results show that expert and Machine Learning (ML) estimate performances are similar, with the experts' performance being slightly better. Both findings confirmed existing literature, but using substantially different experimental settings.**

***Index Terms*—empirical software engineering, software effort estimation, software maintenance issues, machine learning, NLP, BERT, TF–IDF, datasets, Planing Poker**

I. Introduction

Software effort estimation (SEE) plays a critical role in planning activities during the Software Development Life Cycle (SDLC), providing the basis for developing project budgets, schedules and roadmaps. Numerous studies have reported on the difficulties of producing reliable software effort estimates for tasks, resulting in budget over-runs and/or the delivery of low quality software. For instance, Grimstad et al. [1] found that tight schedules give developers a reason for not assuring the quality of their code. Expert-based SEE methods are considered more reliable and popular than approaches based on formal models among practitioners [2, 3]. Nonetheless, expert-based SEE methods are both labour intensive and rely on the availability of suitably experienced estimators.

Considerable attention has therefore been paid in the literature to methods that can automate SEE, reducing costs without impacting accuracy. Both parametric and machine learning (ML) based methods have been explored for this purpose [4, 5]. Unlike parametric methods, ML methods build implicit data models of associations between dependent (effort) and independent variables. ML models thus potentially offers more flexibility in identifying correlations between task features and effort than the predefined associations in parametric methods. Wen et al. [5] reports that in both cases, most studies use numerical features, e.g. the number of source code lines, as a basis for prediction. Aside from a small number of studies, [6, 7, 8, 9], the literature lacks research into the use of a text corpus that describes the software task to predict effort.

Recent advancements in natural language processing (NLP) enable richer feature extraction. In particular, language models such as Bidirectional Encoder Representations from Transformers (BERT) are able to extract features from a text corpus that is aware of the text context [10]. These techniques allow the features of a word to depend on it's context, or in Firth's summary of 1957 the underlying linguistic theory of Distributional Structure, "You shall know a word by the company it keeps!".

Further, these techniques also allow general contextual models of languages to be tuned for application in more specialised domains, i.e. effort estimation. This approach has already been explored by Fávero et al. [8], who used text-based effort estimation along with BERT to extract features from a bugs corpus. Their results indicate the potential for effective effort prediction using this approach. However, Fávero et al.'s study [8] is based on the assumption that pre-trained text representation is better than other feature extraction methods, e.g. Term Frequency–Inverse Document Frequency (TF–IDF).

In the case of advanced NLP language models, e.g. BERT, it is not clear whether improvements in estimation performance is due to better feature extraction from the text corpus, or better classification or regression algorithms for consequent prediction. A second concern is how such NLP advances performance in comparison to expert-based estimates. To evaluate the context-aware and context-less features, this paper uses two feature extraction methods, specifically, TF–IDF as a context-less feature extraction method and BERT as a context-aware feature extraction method. While TF–IDF lacks context properties a word may carry, such as multiple meanings in homonyms and paragraph contexts, it is faster than BERT. In

addition, technical corpus such as the text in issues description may contain content such as stack traces that may undermine the benefits of employing advanced natural language models.

Contribution: In this paper, we therefore address this gap by evaluating effort estimation using Random Forest (RF) and BERT using three open-source datasets and one commercial-based dataset. We also take a deeper investigation of recent NLP advances and evaluate the effects of context-less and context-aware feature extraction on prediction reliability. In addition, we compare expert-based estimates with best performing ML-based predictions.

Previous studies have treated effort estimation as a regression problem, attempting to express effort as numerical value such as person-hours. However, contemporary expert based estimation methods such as Planning Poker [12] express task effort in metaphorical categories that suggest increasing uncertainty with estimate magnitude. For example, Grenning [12] suggests using a Fibonacci sequence of *story points* to indicate the margin of error between estimate sizes as they increase in magnitude. Cohn [3] states that approximate person-effort categories are more appropriate because it is often unrealistic to expect person-hour precision estimates to be accurate for software tasks. Menzies and Shepperd [13] asserts that data discretisation concentrates signals in datasets, which significantly enhances the ML model's performance. The discretisation is an essential consideration since the datasets that are used in this paper have expert-based effort estimates (as baseline). Further, Cohn [3] argues that teams eventually develop a tacit interpretation of the relationship between the relative categorical estimate and actual person-time costs, as the completed tasks are compared to the team's available person-hour budget over several sprints.

In this paper, inspired by the Planning Poker [12] estimation method, we therefore describe and apply a task effort discretisation method that transforms person-time estimates recorded in the employed datasets into categorical estimates. This Planning Poker [12] inspired approach was used in this paper to design a new discretisation category that avoids some drawbacks, such as discretisation noise [14], and brings in the concept of the magnitude of the error to discretisation. Thus, this discretisation is referred to as magnitude discretisation in this paper. This approach models the evaluation of effort estimation after the predominant practice in industry [15] and enables us in to treat effort estimation as a classification problem, rather than regression.

The rest of this paper is structured as follows. Section II presents related work on effort estimation and discusses the contribution made by this paper in more detail. Section III explains the experiment design, including choice of datasets and feature extraction and classification methods. Section IV details the results of three combination between the selected classifiers and feature extraction methods. Then, Section V discusses the results along with validity threats. Finally, Section VII sums up the paper and highlights future research.

II. Related Work

In a survey of effort estimation studies Wen et al. [5] indicates that regardless of algorithmic (parametric or ML) approach used, most studies use numerical features, e.g. the number of source code lines, as a basis for prediction. However, early research has contended that analytical models based on numerical inputs alone are insufficient for achieving improvements in effort estimation [16].

More recent work has begun to consider text based features for effort estimation, such as task description. For example, Ionescu [7] found that features extracted from text offer a more profound linkage between the issues than numeric properties only. Recent advancements in ML and natural language processing (NLP) enable richer feature extraction. In particular, language models such as BERT are able to extract features from a text corpus that is aware of the text context [10]. In addition, general models of language and context can be tuned for specialist domains, i.e. software task descriptions, enabling transfer learning.

Several studies in effort estimation [6, 17, 8] have used different deep learning ML models to predict effort estimation from text-based features at a task level. While none of these studies have compared machine-based with expert based estimates (this paper's baseline), they based their comparisons on actual effort (this paper's ground truth) reported in the datasets selected in their respective studies.

Choetkiertikul et al. [6] were able to produce a better performance of effort estimates for Agile software development projects using their ML models. The dataset collection approach presented in the paper and Choetkiertikul et al. [6]'s are similar. Both approaches source software issues from open-source communities that use JIRA issue tracker system. However, Choetkiertikul et al. [6] didn't utilise the existing pre-trained language models such as BERT, and thus, Choetkiertikul et al. [6] model was trained only on a more limited dataset compared with the results presented in this paper. This therefore reduces the range of language features that may be observed in the task description.

Ardimento and Mele [17] focused on finding the overall time-to fix for bugs in their dataset by utilising the text in the description and developer comments. Similar to this paper, Ardimento and Mele used BERT as a base of their data model. However, the study extracted the corpus from the Bugzilla issue tracking system rather than JIRA, and tuned BERT data model to predict whether a bug resolution will be in one of two categories, slow or fast. The focus of Ardimento and Mele's work was focused on handling speed (slow or fast) compared with more nuanced issue effort categorisation as described in this paper. While predicting the time to fix may help in prioritising issue triage, it may not help in building project budgets and plans.

More recently Fávero et al. [8] used text-based effort estimation along with BERT to extract features from a bug corpus. Their experiment results suggest an effective time prediction. Additionally, Fávero et al. [8] compared two pre-

trained embedding text representations (Word2Vec and BERT) using the DEEP-SE dataset [6], and their experiment resulted in promising outcomes which suggest that using BERT as a contextualised text-embedding representation increases the prediction accuracy. Similar to this study, Fávero et al. used SEEP-SE dataset and tuned BERT to predict the estimates. As mentioned earlier, Fávero et al.'s study has not verified that using pre-trained language models, i.e. BERT, as a feature extraction method was better than contextless methods, i.e. TF–IDF. Unlike Fávero et al.'s evaluation, this paper has used two different classification methods and two feature extraction methods to understand the source of the reported enhancement in Fávero et al.'s evaluation.

III. Experimental Design

This section illustrates the experimental design to evaluate RF and BERT's feed-forward network linear classifiers to predict the effort estimation category. Both classifiers are selected because they fall under the top three classifier categories mentioned by both ML surveys [5, 18]. In addition, the feed-forward network linear classifier is the default classifier that comes with the Transformers library [19]. The experiment also compares two feature extraction methods, BERT embeddings and TF-IDF vectorisation. Finally, it also evaluates prediction performance with expert estimates (baseline) for those projects that have expert estimates. The experiment was designed to answer the following research questions about ML-based SEE methods:

- RQ1: How accurate are the selected ML models in predicting actual effort?
- RQ2: Is there a significant difference between BERT embeddings and TF–IDF vectorisation?
- RQ3: How comparable are the selected ML model predictions to expert estimates?

A. Experiment Datasets

The four datasets selected for the experiment, specifically the JIRA Open Source Software Effort (JOSSE) [20], Planning Poker Industry (PPI), Deep-SE [6], and Porru's [21]. PPI have been collected as part of the research for this paper, and JOSSE, Deep-SE and Porru are publicly available, task-based, annotated with text corpus and effort estimates. In all the datasets, task description is the independent variable and estimate category is the dependent variable. Three datasets are drawn from open-source communities (JOSS, Deep-SE, and Porru's), and PPI was derived from a commercial project. Table Igive summary information about the datasets and their properties. The tables lists the number of records inside each dataset (Deep-SE is the largest), and the minimum, maximum, mean, median, and standard deviation of the dependent variable (effort estimates) in both person-hours and story points.

A subset of the JOSSE dataset has previously been published in a replication pack for the study described by Alhamed and Storer [22], with the full archive now made available on GitHub [20]. The JOSSE dataset [20] was collected from three open-source communities, including Apache, JBoss, and Spring. It consists of 16,979 issues annotated with actual effort, and 4327 issues are annotated with expert-based estimated effort and actual effort in person-hours.

The Deep-SE dataset is derived from Choetkiertikul et al.'s study [6]. They collected their data in the same way as for JOSS, by mining JIRA systems. The data was collected from 9 open-source communities (Apache, Appcelerator, DuraSpace, Atlassian, Moodle, Lsstcorp, MuleSoft, Spring, and Talendforge) and belongs to 17 different projects. Effort in the Deep-SE dataset is represented by story points. While this dataset offers different attributes, e.g. lines of code, only task description and logged effort were extracted for the present study. Porru's dataset was obtained from Porru et al.'s study [21]. It was also collected using the same method as JOSSE and Deep-SE. It consists of 4908 data points collected from 8 open-source projects (Aptana Studio, Dnn Platform, Apache Mesos, Mule, Sonatype's Nexus, Titanium SDK/ CLI, Appcelerator Studio, Spring XD).

The Planning Poker – Industry (PPI) dataset was collected as part of the research for the present paper. The authors has an opportunity to observe how an agile team in an industry partner in the tourism industry. The team were workin on a web application that provides a software as a service function. The authors has an opportunity to observe how the team plays Planning Poker [12] for a period of two weeks. Among the research activities, the researchers was able to collect data about the software development issues of a commercial software product, which included the Planning Poker estimates the team had prepared.

The data was collected from the company issue tracker system (Trello). The issues were classified into different smaller projects or sprints. Two criteria were used to find relevant issues: the actual time spent on the issue and issue status. The team annotated the spent time and the estimated time in the issue's title using square brackets. The data collection took place during an observation period of the team practice of Planning Poker.

The dataset consists of 282 issues that are annotated with actual effort and estimated effort. All the issues have a corpus that is produced by combining the issue title with its description. For every issue, the actual effort and issue features are provided. Expert-estimated effort was also extracted. The issue features consist of a number of comments and a number of activities on the issue. The number of issue activities was extracted from the issue log. It represents a sum of all the events that happened for a given issue. Moreover, the original issue key is used as an identifier in the dataset. Table I present a summary overview of the dataset. While the data points (issues) have been classified into different categories, they all concern the same software, and thus they are treated under one project consisting of 282 data points.

Since PPI was collected from a commercial development house, not permitted the publication of the data since it contains sensitive information that may impact the company interest. As a consequence, the description of the dataset is limited to non-sensitive details. However, use of the dataset

Dataset	# of Projects	# of Records	Has Corpus	Publish Year	# of Expert Estimates (%)	Unit	Min-Max	Avg.	Med.	STD	Skew.
JOSSE	38	16979	yes	2022	4327 (18.7%)	P/H	2-2640	136	60	248	5.06
PPI	N/A	282	yes	N/A	282 (100%)	P/H	5-1560	431	240	416	1.09
Deep-SE	16	23313	yes	2019	0 (0%)	SP	1-100	6	4	10	6.00
Porru	8	4682	yes	2016	0 (0%)	SP	1-6765	5	3	99	68.10

TABLE I: Summary characteristics of experimental datasets. Effort units are shown in person-hour (P/H) or story points (SP). Metrics presented are min/max of estimates in relevant units in the dataset, mean average, median, standard deviation and skewness.

	DEEP-SE	JOSSE	Porru	PPI	# of Records
PPI	0%	0%	0%	100%	272
Porru	13%	0%	100%		4682
JOSSE	0%	100%			16979
DEEP-SE	100%				23313

TABLE II: Pairwise evaluation of duplication between datasets of software tasks used in the study. The table shows the intersection of the two datasets as a proportion of the union of the two sets.

in the study enables comparision of results with open source datasets.

The four datasets were checked for distinctiveness by performing pairwise comparison of issue identifiers. Table II, calculating the proportion of the intersection of the two issue datasets that appeared in the union of the datasets. As can be seen in the results, duplication across datasets was low, with just 13% of tasks duplicated between the Porru and Deep-SE datasets.

B. Dataset Refinement for JOSSE and PPI

The Porru and Deep-SE datasets have already been subject to refinement by the respective study authors. The JOSSE and PPI datasets were subject to refinements as described in the JOSSE Archive [20]. Specifically, Data Points Quantity Check, Outlier Removal, Inconsistency Check and Readability Check. Figure 1 illustrates the four data refinement procedure described in the JOSSE archive [20] and summarized as the following:

- Data Points Quantity Check: data points in the selected datasets belong to different projects, and to avoid treating different contexts of those data point as a one we group the data points for each project and train the ML models on those project individually. However, some projects have as few as one issues, and thus, a 100 data points was determined as a minimum number of data points for each project. Any project with less than a 100 data points were removed.
- Outlier Removal: The dataset's outlier data points are detected using David and Tukey's method [23] to identify data points outside the lower and upper fences. The outlier detection has been done on a project basis, i.e. data points are grouped based on their projects, then the outliers were identified.
- Inconsistency Check: According to Bosu and Macdonell [24], inconsistency is a lack of data point harmony in terms of their property values. Phannachitta et al. [25] state that inconsistency is when the assumption that similar tasks have similar efforts is violated. In other words, if data points have similar effort, they should exhibit similar property values (similar to each other). The challenge with the selected dataset is that the main property is a corpus, and measuring similarity between different corpuses must consider lexical similarity as well as semantic similarity. Thus, BERT embeddings for text corpus were extracted and then a cosine similarity between issue embedding was calculated to determine if there are issues with similar estimates, but with a totally different text corpus. Overall, the datasets are cohesive, that is, the data point properties exhibit high similarity, except for a negligible number of data points per project (less than 5% dissension).
- Readability Check: The data point's corpus contains stack traces or codes without any snippet delimiters, e.g. "<code>"., which makes it difficult to separate such snippets from the descriptions. Language models such as BERT, are sensitive to language errors since they are trained on grammar-free corpuses, and word positions and forms have significance in BERT embeddings. Thus, the corpus for each data point needs to be assessed for language readability. We used the LanguageTool [26] grammar checker to compare the number of grammar errors against the number of corpus words, and based on that, a percentage for the errors can be produced. We determined that any corpus with 22% grammar errors will probably contain a stack trace and was removed. The 22% was the average (8%) plus the stander deviation (14%) grammar errors resulted by running the same grammar checking tool on data-points without stack trace.

The data refinement resulted in a 40.7% in data loss. To test whether this filtering process influenced the profile of the issues selected, the Mean Magnitude of Relative Error (MMRE) of the expert estimates for the filtered and unfiltered datasets was calculated. The MMRE of the filtered dataset was 280.2% compared to 292.3% of the unfiltered dataset. Based on the MMRE delta (12.1%) a decision was made to proceed

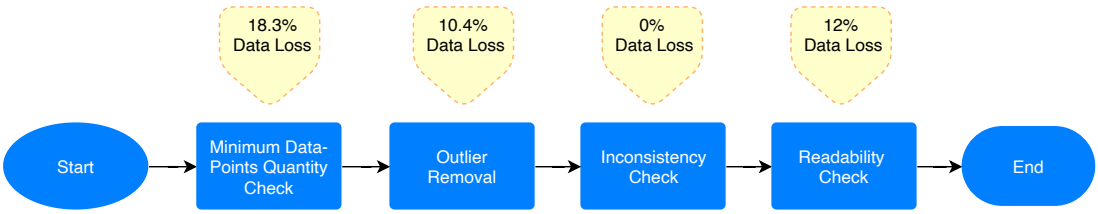


Fig. 1: Dataset Refinement Process as Explained in the JOSSE archive [20].

Category	Low	Middle	High
One hour	0	1	1
Half a day	2	4	5
A day	6	8	10
Half a week	11	20	30
A week	31	40	60
Two weeks	61	80	120

TABLE III: Adopted hour-based categories and their boundaries for a software task effort estimate.

with the filtered version since there appeared to be a minimal impact on the MMREs.

All the datasets are annotated with the actual effort the software development task took. However, datasets use different scales and units. For example, JOSSE is annotated in person-time, whereas Deep-SE is annotated in story points. The actual effort is transformed into time categories with a magnitude scale based on the Fibonacci series. Therefore, the experiment's dependent variable (effort) was discretised to fit those categories. The person-time costs reported on the issues were translated into approximate person-day and person-week categories, labelled as one hour, half a day, one day, half a week, one week, two weeks, and more than two weeks. The translation followed the same scheme as in the community issue tracker system (JIRA), where a working day is equal to 8 hours and a working week equal to 40 hours. This enabled a comparison between predicted estimates and the person-time or story point costs reported on the issues. To draw boundaries between the scale categories, a relative midpoint between the two categories was selected. Table III illustrates the low, middle, and high possible person-hours for each category.

C. Estimation Method

The method used to estimate effort using an ML model has two phases: data preprocessing and model training. The data preprocessing phase is essential for a dataset that has a corpus. During the preprocessing phase, features used to train the ML models are extracted from each issue's corpus. This experiment uses two feature extraction methods, BERT and TF-IDF, as illustrated in Figure 2.

Using the BERT language model as the feature extraction process, the method starts by splitting the text into single words and replacing each word with its corresponding BERT token using the BERT dictionary. Then, the BERT-based [27] pre-training model is used to extract corresponding embeddings of the tokenised corpus as fixed-length vectors. These vectors represent the lexical and semantic meaning of a given word in its context. In this experiment, the embedding process considered only the first 400 words of a given corpus due to limited computing resources available for the study. The March 11th, 2020 version of the BERT-base model was used for the encoding.

TF-IDF was used as an alternative feature extraction method. TF–IDF produces a vector with a length equal to the total number of distinct words in a given corpus. Then, each digit in the vector reflects how many times the corresponding word occurs in the corpus. The count matrix was normalised to avoid the dominant effect of popular words.

After producing BERT embeddings and TF–IDF normalised vectors for the data points, the vectors were sent to the training phase. The training was done on two ML classifiers (RF and BERT linear classifiers). The BERT linear classifier is a single layer of a Feed-Forward Artificial Neural Network (FFANN) on top of BERT. Data points were divided into testing and training subsets using K-Fold Cross-Validation (CV). The training and testing were done on a project basis, which means that datasets with multiple projects, such as JOSSE, were divided into several subsets based on their projects. The Scikit-learn [28] implementation of RF and the McCormick [29] implementation of BERT and its classifier were used to run this experiment. Both models used the default configuration of their original authors [28, 29]. A replication pack of the actual code along with the datasets is publicly available at a GitHub repository[1].

D. Evaluation Metrics

Usually, SEE studies use error measures, e.g. Median Magnitude of Relative Error (MdMRE), as explained in Ali and Gravino's recent survey [18]. Error measures are used as a proxy to performance for a regression ML problem. However, in this paper, the ML task is a classification problem since the data has been discretised, as explained in the previous section (III-B). Thus, accuracy is used as a performance measure of the models.

Two accuracy measures are reported, the Area Under the Curve Receiver Operating Characteristic (AUC-ROC) [30] and F-score [31]. Both metrics are less impacted by imbalanced data points, which happens to be the case for selected datasets.

Those metrics are calculated during model performance testing. The testing method used is K-Fold Cross-Validation

[1]https://github.com/ml-see/replication-pack

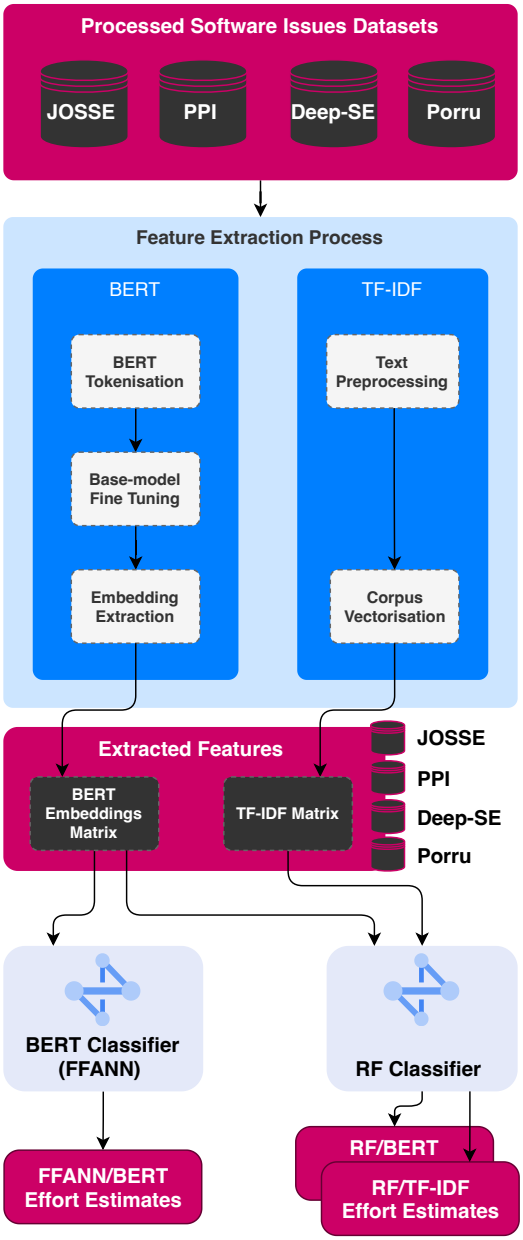


Fig. 2: Overview all the feature extraction methods (BERT and TF–IDF), classifiers (FFANN and RF), datasets (JOSSE, PPI, DEEP-SE and Porru)

(CV) [32]; a five-fold CV is implemented. In each fold, four-fifths of the data is used for training and one fifth is used for testing. Statistical tests of their significance are carried out using Kruskal-Wallis tests and ANOVA for AUC-ROC and F-score, respectively, at a significance level of 0.05.

IV. Results

This section details the results of predicting estimates using FFANN and RF classifiers. The first part gives the prediction performance measurements using one feature extraction method, BERT, across two classifiers, RF and FFANN, to assess the impact of the classifier on accuracy. The second part

Dataset	ML Classifier	Feature Extraction	Projects	Folds	FS	AR
Deep-SE	FFANN	BERT	7	68	0.434	0.633
	RF	BERT	7	68	0.351	0.613
	RF	TF-IDF	7	68	0.361	0.585
JOSSE	FFANN	BERT	35	167	0.680	0.561
	RF	BERT	35	167	0.612	0.551
	RF	TF-IDF	35	167	0.657	0.537
Porru	FFANN	BERT	4	17	0.404	0.571
	RF	BERT	4	17	0.314	0.556
	RF	TF-IDF	4	17	0.336	0.578
PPI	FFANN	BERT	1	5	0.502	0.618
	RF	BERT	1	5	0.388	0.604
	RF	TF-IDF	1	5	0.616	0.754

TABLE IV: Results of different models across different datasets. The results for JOSSE, Deep-SE, and Porru are aggregated from results of individual projects inside each dataset. Mean is used as the aggregation function. FS stands for F-Score and AR stands for AUCROC

evaluates feature extraction methods by using an RF classifier across two feature extraction methods. As stated earlier in the introduction, the aim is to examine BERT as a transfer learning model in the SEE problem, and thus BERT is compared with TF-IDF using the same classifier. Finally, to put the results in context, the third part compares ML models with expert estimates to identify more meaningful aspects from an expert estimation point of view.

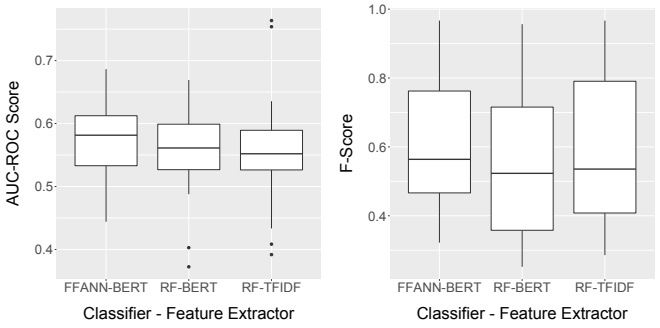
Table IV shows the summarised results across the datasets. The performance measures of F-Score and AUC-ROC are aggregated by averaging across all projects in the dataset. For more detailed results, the appendix lists the performance metrics based on individual projects for each dataset and can be accessed using the same replication pack repository[1].

A. *RQ1: Accuracy of ML models*

The results given in Table IV suggest that using BERT for feature extraction and BERT's linear classifier (FFANN) for classification is slightly better than the other options. Across all datasets, the FFANN-BERT combination achieved better F-Score and AUC-ROC results than other combinations (RF-BERT and RF-TF-IDF), except for the PPI dataset, for which RF-TF-IDF performed better.

However, we have investigated the performance on a project level using Kruskal–Wallis for AUC-ROC and ANOVA for F-Score. The Kruskal–Wallis test results in 2.92 with a p-value of 0.232 for AUC-ROC, and the ANOVA test results in 1.668 with a p-value of 0.192. Both tests for both metrics show no significant difference between the three combinations, and thus, the null hypothesis cannot be rejected. Both classifiers have similar accuracy performance using BERT and TF-IDF feature extraction methods.

To visualise the performance metrics, Figure 3 shows both AUC-ROC (3a) and F-Score (3b) for the three combinations using box plots. The figure illustrates the slight, but not significant, difference between them.



(a) Box Plot of AUC-ROC (b) Box Plot of F-Score

Fig. 3: Box plot of performance metrics for different models/feature extraction methods.

Project	Number of Issues	F-Score
AEROGEAR	185	0.678
BATCH	290	0.813
EXOJCR	489	0.536
GTNPORTAL	166	0.713
INT	385	0.843
JBTM	114	0.554
MNG	113	0.723
RF	236	0.786
STDCXX	178	0.637
	F-Score Mean	0.698

TABLE V: F-Score of expert estimates reported for selected JOSSE projects.

The AUC-ROC metric gives an overall accuracy measurement of a classifier for different classification probability thresholds. Its accuracy measurement focuses on classifier specificity. Classifier specificity measures how good the classifier is in identifying data points associated with a negative class. A classifier with 100% specificity means that it never misses a negative data point. Figure 3a shows that FFANN-BERT combination did the best. BERT as a feature extraction method also performed better than TF-IDF. Only two projects, PPI and JBEAP (outliers), performed well using the RF-TF-IDF combination. While these are slight differences, there is no statistically significant difference.

On the other hand, F-Score metric gives an accuracy measurement of a classifier based on maximum classification probability (a single threshold). Unlike AUC-ROC, F-Score focuses on classifier precision. Classifier precision measures how good the classifier is in identifying data points with a positive class. A classifier with 100% precision means that it never misses a positive data point. Figure 3b shows that the FFANN-BERT and RF-TF-IDF combinations performed similarly. This time, BERT as a feature extraction method performed worse than TF-IDF. Although all the projects are software development projects, the noticeably wider IQR of F-Score indicates that projects still vary in the association between task descriptions and time logging.

While these are slight differences, there is no statistically significant difference. For a multi-class problem, as in the case of the SEE problem, a strategy is used which pits one class against the rest of the classes, and then the average of the classes' metrics is calculated as a summary metric.

B. RQ2: Evaluation of Feature Extraction Methods

To examine whether any of the classifiers had impact on the accuracy performance, a narrow comparison between two combinations of FFANN-BERT and RF-BERT was performed where the only feature extraction method used was BERT. Applying a statistical significance test of Kruskal–Wallis on the AUC-ROC scores of all projects, resulted in 1.1288 with a p-value of 0.29. Similarly, applying ANOVA on F-Score resulted in 3.34 with a p-value of 0.071. Both tests on both metrics indicate that there is no significant difference, and thus the classifiers have no impact on the accuracy performance.

Next, a comparison between the two feature extraction methods using an RF classifier (RF-BERT and RF-TF-IDF) is performed. RF was selected since FFANN is built upon BERT and its implementation expects BERT-format input, whereas the RF implementation accepts both TF-IDF and BERT matrices. The aim is to see whether BERT embeddings built upon a large language model will result in a significant difference compared with a context-less feature extraction such as TF-IDF.

Applying the statistical significance test of Kruskal–Wallis on the AUC-ROC scores of all projects resulted in 0.505 with a p-value of 0.48. Similarly, applying ANOVA on F-Score resulted in 1.23 with a p-value of 0.27. Both tests on both metrics indicate that there is no significant difference, and thus BERT embeddings are not necessarily better than TF-IDF for the SEE problem.

C. RQ3: ML models compared with expert-based estimates

To put those performance metrics in context, the F-Score of the expert-based estimates reported in the JOSSE dataset was calculated. Table V shows the performance of the expert-based estimates for individual projects of JOSSE.

The average F-Score is 0.7, with the best performance being 0.812 and the worst being 0.54. Comparing these scores with the best performing ML model, expert estimates are better than the ML-based estimates. The ANOVA test of F-Score for those projects resulted in 4.685 with a p-value of 0.046, indicating a significant difference between expert and ML-model estimates.

Three projects, BATCH, EXOJCR, and INT, have a noticeably large number of issues (290, 489, 385 respectively). Two of them (BATCH and INT) achieved the best F-Scores (0.813 and 0.843). This might shed light on the practice of effort estimation in open-source projects. In addition, the percentage of issues annotated with expert estimates [20] in those projects is high (BATCH: 89.4% and 93.2%). This may indicate that those projects approached effort estimation rigorously, which may have helped in achieving higher F-Scores.

To explore this conjecture, ten members of the communities were contacted to determine how the estimates were conducted. Only five of them responded and indicated that

there was no particular procedure or instruction around effort estimations. One respondent stated that the development team tries to experiment with different SEE methods but always relying on their "gut feeling". That respondent was a tester. The respondent takes the following considerations while estimating time for a software development task:

- The required learning about the development task and its deployment.
- Manual versus automated deployment.
- Reproducing the software issue and creating a fix.
- Creation of test cases.
- Team members scheduled and holidays.
- Contingency time.

Further, the respondent commented that learning about the development task and its deployment may take a long time depending on the difficulty of the system environments. Other respondents explained that they rely on their experience when estimating the issues, with or without a structured process for predicting such estimates.

V. Discussion

This section discusses the results from different perspectives and compares them with relevant literature. Threats to validity are also discussed and proposes a hybrid approach to automated effort estimation that retains the human-in-the-loop as a next step for advancing SEE.

As presented in the results, the lack of significant difference between both classifiers in accuracy performance is also reported by previous studies, such as the systematic literature review of Wen et al. [5]. According to the review, the average of the Mean Magnitude of Relative Error (MMRE) for DT-based models in 17 experiments was 55%, whereas it was 37% for ANN-based models reported in 39 experiments. While ANN-based models performed better, the difference is not significant as reported earlier in Section IV in this paper. Nonetheless, the reported metrics cannot be compared with those in this paper, since the SEE problem in this paper is a classification problem, whereas for the above survey it was a regression problem. However, the improvement delta reported in the survey agrees with what is reported in this paper.

From a different but close perspective, ensemble methods using DT-based models ensemble significantly outperform ensemble methods using ANN-based models, as reported by Idri et al.'s survey [33]. The average of MMRE for DT-based ensemble in 5 experiments was 17.57%, whereas it was 48.32% for ANN-based ensemble reported in 16 experiments. Since Idri et al.'s survey [33] contradicts the trend reported by Wen et al. [5], that weakens the overall difference between the models.

Datasets are another factor, other than model algorithms, that may affect accuracy performance. Thus, the datasets included in this experiment were taken through a series of refinement steps, starting with ensuring an adequate number of data points for each project (at least 100 data points). Then, outlier data points were removed. Inconsistent or unreadable data points (i.e. with code or stack trace) were checked and removed (refer to JOSSE's paper [20] for more details). This helped to enhance BERT classifier accuracy performance by 16%. However, all performance measures are close to a random prediction measure. Theoretically, random predictions have a measure of 0.5 on both F-Score and AUC-ROC. The reported results of the datasets shown in Table IV are close to random prediction, with RF-TF-IDF using the PPI dataset being the only exception.

Interestingly, TF-IDF shows slightly better performance using the RF classifier over FFANN, which is not expected, since FFANN relies on a state-of-the-art language model, i.e. BERT. This can indicate that language-based models do not necessarily offer the best feature extraction method for non-language applications, as in effort estimation.

Another reason behind BERT's lack of performance is the technical nature of the language of the task descriptions. Although such models have been pre-trained on proper English content and fine-tuned using domain-specific datasets such as JOSSE, software issues may not be written in proper English, as explained in the previous JOSSE's paper. Technical writing may lack proper English grammar and sentence structure. Nonetheless, BERT embeddings achieved the best performance when used with BERT's linear classifier (FFANN).

VI. Threats to Validity

A possible threat to validity is that the data are collected from open-source projects, where time control and project management are more relaxed compared with commercial projects. Thus, time logging for task effort and expert estimates might not follow a specific protocol or process, as explained in Question 3's answer.

The threat to validity that may originate from the filtrating process lowering the generality of the filtered version of the dataset. It is possible that the filtering process gave advantages to the expert estimates by excluding bad expert estimates during the filtering process. Therefore, the MMRE of both datasets versions, filtered and unfiltered, were 280.2% and 292.3% respectively. The delta between the MMRE indicates that expert estimates in both versions are similar with a neglectable difference.

To mitigate this risk, a commercial-project dataset was collected and included in the comparison, where the development team followed a defined effort estimation and logging method, i.e. Planning Poker. In addition, the discretisation step is designed to minimise the risk in two ways. Firstly, it reduces the granularity of time to larger time buckets, and secondly, the efforts compared are in discrete, not absolute, values.

Changes to expert estimates in the JOSSE dataset after realising the actual effort is a threat that is mitigated by reviewing the activity log of software issues with expert estimates. None of them had a log entry indicating that the field $timeestimate$ has been changed after updating the field $timespent$. $timeestimate$ and $timespent$ fields correspond to the expert estimate and actual time respectively.

Conclusions about accuracy performance can be threatened by the nature of the datasets. For instance, imbalanced datasets

have artificial effects on some accuracy scores. To minimise such a threat, the measures F-score and AUC-ROC, which can handle imbalanced data, were selected. In addition, the statistical tests of Kruskal–Wallis and ANOVA were used to draw final conclusions.

Threats to external validity is also mitigated by incorporating issues from different large software projects, including both industrial and open-source projects. While the issues vary, they do not represent all kinds of software. For instance, critical system software projects, where the tightest time management is expected, are not included.

VII. Conclusion

This paper reflects upon recent research in the area of ML SEE. It focuses on ML methods since they are the most researched methods in the recent literature [5]. ML methods are sensitive to the data that they are trained on, and thus datasets represent the other half of ML SEE research.

It draws a comparison between an ensemble model (RF) and a pre-trained language model (BERT) regarding their performance and feature extraction methods in an experimental study. It used four datasets, specifically JOSSE, PPI, Deep-SE, and Porru. It also used expert estimates in the JOSSE dataset to compare ML models with experts.

The results suggest that there is no significant difference between the presented methods. However, BERT-BERT shows slightly better performance. On the other hand, the results show that expert and ML estimate performances are similar, with the experts' performance slightly better. Both findings confirmed what was already reported in the literature using different experimental settings [34, 35].

While the expert estimations are significantly better than ML, they are still unreliable, and there is room for improvement. Perhaps by combining both ML and expert in one estimation method, each could strengthen the other.

Future work for ML-based effort estimation studies may benefit from involving human judgement in the ML process. Effort estimation is not a trivial process, especially for intangible deliverables like software. Neither text-based features nor project characters are enough to build a reliable data model. Perhaps involving humans in the loop may help in comprehending estimation complexity.

References

[1] S. Grimstad, M. Jørgensen, and K. Moløkken-Østvold, "Software effort estimation terminology: The tower of Babel," *Inf. Softw. Technol.*, vol. 48, no. 4, pp. 302–310, 2006. [Online]. Available: https://doi.org/10.1016/j.infsof.2005.04.004

[2] M. Usman, E. Mendes, and J. Börstler, "Effort estimation in agile software development: a survey on the state of the practice," in *Proceedings of the 19th International Conference on Evaluation and Assessment in Software Engineering, EASE 2015, Nanjing, China, April 27-29, 2015*, J. Lv, H. J. Zhang, and M. A. Babar, Eds. ACM, 2015, pp. 12:1–12:10. [Online]. Available: https://doi.org/10.1145/2745802.2745813

[3] M. Cohn, *Agile Estimating and Planning*, ser. Robert C. Martin Series. Pearson Education, 2005.

[4] B. W. Boehm, "Software engineering economics," *Software Pioneers*, p. 641686, 2002. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-59412-0_38

[5] J. Wen, S. Li, Z. Lin, Y. Hu, and C. Huang, "Systematic literature review of machine learning based software development effort estimation models," *Inf. Softw. Technol.*, vol. 54, no. 1, pp. 41–59, 2012. [Online]. Available: https://doi.org/10.1016/j.infsof.2011.09.002

[6] M. Choetkiertikul, H. K. Dam, T. Tran, T. Pham, A. Ghose, and T. Menzies, "A deep learning model for estimating story points," *IEEE Trans. Software Eng.*, vol. 45, no. 7, pp. 637–656, 2019. [Online]. Available: https://doi.org/10.1109/TSE.2018.2792473

[7] V. Ionescu, "An approach to software development effort estimation using machine learning," in *13th IEEE International Conference on Intelligent Computer Communication and Processing, ICCP 2017, Cluj-Napoca, Romania, September 7-9, 2017*. IEEE, 2017, pp. 197–203. [Online]. Available: https://doi.org/10.1109/ICCP.2017.8117004

[8] E. M. Fávero, D. Casanova, and A. R. Pimentel, "Se3m: A model for software effort estimation using pre-trained embedding models," *arXiv preprint arXiv:2006.16831*, 2020.

[9] P. Abrahamsson, I. Fronza, R. Moser, J. Vlasenko, and W. Pedrycz, "Predicting development effort from user stories," in *Proceedings of the 5th International Symposium on Empirical Software Engineering and Measurement, ESEM 2011, Banff, AB, Canada, September 22-23, 2011*. IEEE Computer Society, 2011, pp. 400–403. [Online]. Available: https://doi.org/10.1109/ESEM.2011.58

[10] "Open sourcing bert: State-of-the-art pre-training for natural language processing," Nov 2018. [Online]. Available: https://ai.googleblog.com/2018/11/open-sourcing-bert-state-of-art-pre.html

[11] J. R. Firth, "A synopsis of linguistic theory, 1930-1955," *Studies in Linguistic Analysis (special volume of the Philological Society)*, vol. 1952-59, pp. 1–32, 1957.

[12] J. Grenning, "Planning poker or how to avoid analysis paralysis while release planning," *Hawthorn Woods: Renaissance Software Consulting*, vol. 3, pp. 22–23, 2002.

[13] T. Menzies and M. J. Shepperd, "Special issue on repeatable results in software engineering prediction," *Empir. Softw. Eng.*, vol. 17, no. 1-2, pp. 1–17, 2012. [Online]. Available: https://doi.org/10.1007/s10664-011-9193-5

[14] G. K. Rajbahadur, S. Wang, Y. Kamei, and A. E. Hassan, "Impact of discretization noise of the dependent variable on machine learning classifiers in software engineering," *IEEE Trans. Software Eng.*, vol. 47, no. 7, pp. 1414–1430, 2021. [Online]. Available:

https://doi.org/10.1109/TSE.2019.2924371
[15] CollabNet VersionOne, "13th annual state of agile report," https://www.stateofagile.com, May 2019.
[16] T. Mukhopadhyay, S. S. Vicinanza, and M. J. Prietula, "Examining the feasibility of a case-based reasoning model for software effort estimation," *MIS Quarterly*, vol. 16, no. 2, p. 155, Jun 1992. [Online]. Available: http://dx.doi.org/10.2307/249573
[17] P. Ardimento and C. Mele, "Using bert to predict bug-fixing time," *2020 IEEE Conference on Evolving and Adaptive Intelligent Systems (EAIS)*, May 2020. [Online]. Available: http://dx.doi.org/10.1109/eais48028.2020.9122781
[18] A. Ali and C. Gravino, "A systematic literature review of software effort prediction using machine learning methods," *J. Softw. Evol. Process.*, vol. 31, no. 10, 2019. [Online]. Available: https://doi.org/10.1002/smr.2211
[19] "Transformers." [Online]. Available: https://huggingface.co/transformers/index.html
[20] M. Alhamed and T. Storer, "JOSSE: A software development effort dataset annotated with expert estimates," 2022. [Online]. Available: https://github.com/ml-see/josse
[21] S. Porru, A. Murgia, S. Demeyer, M. Marchesi, and R. Tonelli, "Estimating story points from issue reports," in *Proceedings of the The 12th International Conference on Predictive Models and Data Analytics in Software Engineering, PROMISE 2016, Ciudad Real, Spain, September 9, 2016*. ACM, 2016, pp. 2:1–2:10. [Online]. Available: https://doi.org/10.1145/2972958.2972959
[22] M. Alhamed and T. Storer, "Playing planning poker in crowds: Human computation of software effort estimates," in *43rd IEEE/ACM International Conference on Software Engineering, ICSE 2021, Madrid, Spain, 22-30 May 2021*. IEEE, 2021, pp. 1–12. [Online]. Available: https://doi.org/10.1109/ICSE43902.2021.00014
[23] F. N. David and J. W. Tukey, "Exploratory data analysis," *Biometrics*, vol. 33, no. 4, p. 768, Dec 1977. [Online]. Available: http://dx.doi.org/10.2307/2529486
[24] M. F. Bosu and S. G. Macdonell, "Experience: Quality benchmarking of datasets used in software effort estimation," *Journal of Data and Information Quality*, vol. 11, no. 4, 2019.
[25] P. Phannachitta, J. Keung, K. E. Bennin, A. Monden, and K. Matsumoto, "Filter-inc: Handling effort-inconsistency in software effort estimation datasets," *2016 23rd Asia-Pacific Software Engineering Conference (APSEC)*, 2016. [Online]. Available: http://dx.doi.org/10.1109/apsec.2016.035
[26] "Spell and grammar checker." [Online]. Available: https://languagetool.org/
[27] I. Turc, M.-W. Chang, K. Lee, and K. Toutanova, "Well-read students learn better: On the importance of pre-training compact models," *arXiv preprint arXiv:1908.08962v2*, 2019.
[28] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.
[29] C. McCormick, 2021. [Online]. Available: https://www.chrismccormick.ai/the-bert-collection
[30] T. Fawcett, "An introduction to ROC analysis," *Pattern Recognit. Lett.*, vol. 27, no. 8, pp. 861–874, 2006. [Online]. Available: https://doi.org/10.1016/j.patrec.2005.10.010
[31] Y. Sasaki *et al.*, "The truth of the f-measure. 2007," 2007.
[32] M. Stone, "Cross-validatory choice and assessment of statistical predictions," *Journal of the Royal Statistical Society: Series B (Methodological)*, vol. 36, no. 2, p. 111133, Jan 1974. [Online]. Available: http://dx.doi.org/10.1111/j.2517-6161.1974.tb00994.x
[33] A. Idri, M. Hosni, and A. Abran, "Systematic literature review of ensemble effort estimation," *J. Syst. Softw.*, vol. 118, pp. 151–175, 2016. [Online]. Available: https://doi.org/10.1016/j.jss.2016.05.016
[34] A. Zakrani, A. Idri, and M. Hain, "Software effort estimation using an optimal trees ensemble: An empirical comparative study," *Proceedings of the 8th International Conference on Sciences of Electronics, Technologies of Information and Telecommunications (SETIT18), Vol.1*, p. 7282, Jul 2019. [Online]. Available: http://dx.doi.org/10.1007/978-3-030-21005-2_7
[35] M. Jørgensen, B. W. Boehm, and S. Rifkin, "Software development effort estimation: Formal models or expert judgment?" *IEEE Softw.*, vol. 26, no. 2, pp. 14–19, 2009. [Online]. Available: https://doi.org/10.1109/MS.2009.47