Agile Effort Estimation: Have We Solved the Problem Yet? Insights From the Replication of the GPT2SP Study

Vali Tawosi
Department of Computer Science
University College London (UCL)
London, United Kingdom
vali.tawosi@ucl.ac.uk

Rebecca Moussa
Department of Computer Science
University College London (UCL)
London, United Kingdom
r.moussa@ucl.ac.uk

Federica Sarro
Department of Computer Science
University College London (UCL)
London, United Kingdom
f.sarro@ucl.ac.uk

***Abstract*—Replication studies in Software Engineering are indispensable for ensuring the reliability, generalizability, and transparency of research findings. They contribute to the cumulative growth of knowledge in the field and promote a scientific approach that benefits both researchers and practitioners. In this article, we report our experience replicating a recently published work proposing a Transformer-based approach for Agile Story Point Estimation, dubbed GPT2SP.**

GPT2SP was proposed with the intent of addressing the three limitations of a previous Deep Learning-based approach dubbed Deep-SE, and the results reported in the original study set GPT2SP as the new state-of-the-art.

However, when we used the GPT2SP source code made publicly available by the authors of the original study, we found a bug in the computation of the evaluation measure and the re-use of erroneous results from previous work, which had unintentionally introduced biases in the GPT2SP's performance evaluation.

In this study, we report on the results we obtained after fixing the issues present in the original study, which reveal that their results were in fact unintentionally inflated due to these issues and that despite advancements, challenges remain in providing accurate effort estimations for agile software projects.

I. Introduction

Effort estimation in software development has been a longstanding challenge, and recent advancements in machine learning, particularly neural networks, have shown promise in improving the accuracy and reliability of estimation models.

Fu and Tantithamthavorn [1] have recently proposed GPT2SP, a Transformer-based deep learning model for Story Point (SP) estimation of user stories to overcome the limitations of the method previously proposed by Choetkiertikul et al. [2], Deep-SE, which was not accurate enough [3].

They empirically assessed the performance of GPT2SP on the same dataset where Choetkiertikul et al. [2] evaluated Deep-SE, including 16 projects with a total of 23,313 issues. They benchmarked GPT2SP against two baselines (namely the naive Mean Effort and Median Effort) and Deep-SE for both within- and cross-project estimation scenarios. Their results showed that GPT2SP outperforms Deep-SE with a 6%-47% improvement (computed based on the Mean Absolute Error) for the within-project scenario and a 3%-46% improvement for the cross-project scenario. However, when we attempted to use the GPT2SP source code made available by Fu and Tantithamthavorn [1] to reproduce their experiments, we found a bug in the computation of the Mean Absolute Error (MAE), which have inflated the GPT2SP's accuracy reported in their work. Therefore, we issued a pull request fixing such a bug, which was accepted by Fu and Tantithamthavorn and merged into the original work's repository.[1] Furthermore, the original GPT2SP study used the Mean Effort and Median Effort baseline results from the empirical study by Choetkiertikul et al. [2], which were not correct (see Tawosi et al. [3] for a description of the error in the baseline MAE values and for its resolution).

In this paper, we report on the results we have obtained by using the fixed version of GPT2SP and the correct Mean and Median baseline results [3] to replicate the experiments conducted in the original study by Fu and Tantithamthavorn [1].

Following the original study, we analyse GPT2SP in two scenarios, namely within-project and cross-project scenarios, and benchmark its performance with those achieved by three other approaches, namely Deep-SE, Mean Effort, and Median Effort. We evaluate the results based on the Mean Absolute Error (MAE) of the estimation methods over all issues in each project, but also report the Median Absolute Error (MdAE) and the Standard accuracy (SA), for completeness.

Our results showed that, in the within-project scenario, GPT2SP outperforms the Median baseline and Deep-SE in only six cases out of 16, where the difference is statistically significant in only three cases against the Median (two with negligible and one with small effect size), and two cases against Deep-SE (both with negligible effect size). With regards to the cross-project scenario, GPT2SP outperformed Deep-SE in the majority of the cases, however, it performed poorly against the Median baseline (outperforming it statistically significantly in only five of the 16 cases).

[1] https://github.com/awsm-research/gpt2sp/pull/2

Overall, our replication study confirms that GPT2SP performs better than Deep-SE in the cross-repository scenario as concluded by the original study. However, we found that GPT2SP performs poorly when compared against the Median baseline in both scenarios. Therefore, our replication results do not support the main conclusion made in the original study stating that GPT2SP significantly outperforms the baselines.

Based on the findings reported herein and those from a previous replication we carried out on the use of Deep Learning for Agile software effort estimation [3], we conclude that the two approaches proposed in the literature thus far (namely, Deep-SE and GPT2SP [1], [2]) are not more effective than much simpler baselines (such as using the mean or median effort of past user stories as a prediction of the effort required by new user stories) for agile open-source software development effort estimation. Future studies need to further explore and devise suitable techniques that can lead practitioners to obtain more accurate agile software development estimates.

The remainder of the paper is structured as follows. Section II provides an introduction to the estimation models used in this study, including GPT2SP, Deep-SE, and the baseline methods, describes the research questions, the dataset, and measures used to evaluate the models' performance. Section III reports the results of our replication, while Section IV discuss possible threats to validity of our study. Section V covers the related work, and Section VI discusses the main take-away messages and concludes the paper.

II. Methodology

This section provides a brief introduction to GPT2SP and the benchmark techniques investigate in this work the research questions we replicate from the work by Fu and Tantithamthavorn [1], and the datasets and measures we used for evaluation.

A. Techniques

1) GPT2SP: GPT2SP [1] is a Transformer-based deep learning model for SP estimation of user stories, which leverages GPT-2 pre-trained language model. Therefore, GPT2SP introduces three structural and design improvements over Deep-SE. First, instead of word-level tokenization, GPT2SP employs a byte-pair-encoding subword tokenization which splits rare words into subword units, reducing the vocabulary size to almost one-fourth. Second, unlike Deep-SE, which needed pre-training on project-specific data, GPT2SP uses a pre-trained model that can generate meaningful embedding for any project. And third, GPT2SP uses GPT-2 architecture [4] with a masked multi-head self-attention mechanism [5], allowing it to capture the relationship among words better while considering the context of a given word and its position in the sequence. After the GPT-2 model for user story embedding, GPT2SP uses a 3-layer Multi-Layer Perceptron (MLP) regressor to learn a mapping between the embeddings and story point value of the user stories.

2) Deep-SE: Choetkiertikul et al. [2] proposed Deep-SE as a deep learning model to estimate the story point. Their model is composed of four components: (1) Word Embedding, (2) Document representation using Long-Short Term Memory (LSTM) [6], [7], (3) Deep representation using Recurrent Highway Network (RHWN) [8], and (4) Differentiable Regression. The first component converts each word in the title and description of issues (i.e., user story) into a fixed-length vector (i.e., word embedding). These word vectors serve as an input sequence to the LSTM layer, which computes a vector representation for the whole document. The document vector is then fed into the Recurrent Highway Network (RHWN), which transforms the document vector multiple times before outputting a final vector which represents the text. The vector serves as input to the regressor that predicts the story point.

3) Baselines: The original study includes *Mean Effort and Median Effort* as baselines, although these were not specifically considered in their RQs. Mean Effort and Median Effort are two benchmarks commonly used for effort estimation techniques [9]–[11]. Specifically, the mean (or median) story point of the past issues is used as the predicted story point for a new issue, when Mean Effort (or Median Effort) is used.

B. Research Questions

The original study formulated the following three research questions to evaluate their proposal GPT2SP: Does our GPT2SP outperform Deep-SE for within-project scenario?; Does our GPT2SP outperform Deep-SE for cross-project scenarios?; What are the contributions of the components of GPT2SP?.

The within-project estimation uses previous user stories from the same project to train a model and predict for future user stories. Therefore, this scenario is not applicable to a new project that has no user stories yet (known as the cold-start problem). In such a case, a model is trained on user stories from one or more previous projects and used to estimate user stories of the new project (i.e., cross-project estimation).

In our replication study, we answer the first two questions as well as benchmark GPT2SP with three common baselines in effort estimation studies, as this is a best practice and our previous work showed that DeepSE was not able to always outperform simple baselines, thus it is crucial to assess whether this is also the case for GPT2SP [11]. The research questions we answer in this study are as follows:

- RQ1. Does GPT2SP outperform Deep-SE and the baselines for the within-project scenario?
- RQ2. Does GPT2SP outperform Deep-SE and the baselines for the cross-project scenario?

The original study also investigated a third research question: What are the contributions of the components of GPT2SP?. However, We do not replicate this question since the replication of the first two questions showed that GPT2SP is not an effective approach for the task at hand (see Section III), and as such it became out of scope to further investigate the contributions of the components of GPT2SP.

C. Dataset

We use the same benchmark dataset adopted in the original study and therefore we conduct our experiments with 23,313

issues from 16 different projects. This benchmark dataset was originally collected from JIRA and made publicly available by Choetkiertikul et al. [2], [12]. The original study reused it as-is. Specifically, For each project, issues and related key information (i.e., issue ID, title, descriptions, and story points) were collected through JIRA REST API up until August 8, 2016. Table I summarises the statistics of the datasets.

D. Evaluation Measures and Statistical Analysis

Several measures have been used in the software effort estimation literature to measure the accuracy of the estimation models. These measures are generally built upon the error (or absolute error) between the predicted value and the actual value (i.e., $|Actual.value - Predicted.value|$).

Similarly to the original study, we discuss all results of our study based on the Mean Absolute Error (MAE) measure, while we also report the Median Absolute Error (MdAE) and Standard Accuracy (SA) values for completeness.

These measures (defined in Equations 1, 2 and 3) are standardised measures which are not biased towards under- or over-estimates, and thus, recommended by the previous work [13], [14].

Across n issues, the MAE and MdAE are computed as follows:

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|actual_i - predicted_i| \tag{1}$$

$$MdAE = Median_{i=1}^{n}\Big\{|actual_i - predicted_i|\Big\} \tag{2}$$

where $actual_i$ is the actual effort from the historical data, $predicted_i$ is the predicted effort by the method and n is the number of issues in a given project.

The bug in GPT2SP's original implementation affected the computation of MAE, and it arose from dividing the sum of absolute errors by a number greater than n. To be specific, the original implementation processed the input data in batches and computed n as the number of batches multiplied by the length of the first batch, while the way the batches were divided (30% of the set) always left the final batch almost empty with a few issues. This computation produced a number n', which was always greater than the actual value of n. Therefore, the MAE computed was always smaller than the actual MAE.

SA is recommended as a standard measure to compare multiple prediction models against each other [13]. It is based on MAE and is defined as follows:

$$SA = \left(1 - \frac{MAE_{p_i}}{MAE_{p_0}}\right) \times 100 \tag{3}$$

where MAE_{p_i} is the MAE of the approach p_i being evaluated and MAE_{p_0} is the MAE of a large number (usually $1,000$ runs) of *random guesses*.

For a prediction model p_i which outperforms random guessing in terms of accuracy, SA will produce a number in the range $[0, 1]$. An SA value closer to zero means that the predictor p_i is performing just a little better than random guessing [13], [15]. For a prediction model which is outperformed by random guessing SA will produce a negative value. For a high-performance prediction model MAE and $MdAE$ should be lower and SA should be higher than the competitors.

To check if the difference in the results achieved by the two methods is statistically significant, we performed the Wilcoxon Ranked-Sum test (a.k.a. Mann–Whitney U test) on the distribution of the absolute errors produced by the methods under investigation. Specifically, we used a one-sided Wilcoxon test with a confidence limit of $\alpha = 0.05$ to check the following Null Hypothesis:

Null Hypothesis *The distribution of absolute errors produced by GPT2SP is better (i.e., lower) than that produced by prediction model P_i.*

If the test rejects the Null Hypothesis, the alternative hypothesis would be accepted:

Alternative Hypothesis *The distribution of absolute errors produced by GPT2SP is not better (i.e., not lower) than that produced by prediction model P_i.*

We decided to use a non-parametric test such as the Wilcoxon one, as it does not make any assumption on the normality of the distribution of the data at hand. We performed a one-sided test since we are interested in knowing if a given model (i.e., GPT2SP) would commit a smaller estimation error than another model. In such a case, the one-sided p-value interpretation would be straightforward.

To mitigate the risk of incorrectly rejecting the Null Hypothesis (i.e., Type I error) [16], we also analyse how the results would be when the Bonferroni correction is applied to cater for multiple hypothesis testing (i.e., the confidence limit is set as α/K, where K is the number of hypotheses). Therefore, herein we report the original p-value results of the Wilcoxon test.

We also use a standardised non-parametric effect size measure (i.e., the Vargha Delaney's $\hat{A}_{12}$ statistic) to assess the practical magnitude of the difference between two methods, as recommended in previous work [15]–[17]. For two algorithms A and B, the $\hat{A}_{12}$ measures the probability of A performing better than B with respect to a performance measure. $\hat{A}_{12}$ is computed using Equation (4), where R_1 is the rank sum of the first data group being compared, and m and n are the number of observations in the first and second data sample, respectively.

$$\hat{A}_{12} = \frac{\left(\frac{R_1}{m} - \frac{m+1}{2}\right)}{n} \tag{4}$$

Based on Equation (4), if two algorithms are equally good, $\hat{A}_{12} = 0.5$. Respectively, $\hat{A}_{12}$ higher than 0.5 signifies that the first algorithm is more likely to produce better predictions. The effect size is considered negligible for $\hat{A}_{12} < 0.6$ (represented by an 'N'), small (S) for $0.6 \leq \hat{A}_{12} < 0.7$, medium (M) for $0.7 \leq \hat{A}_{12} < 0.8$, and large (L) for $\hat{A}_{12} \geq 0.8$, although these thresholds are not definitive [15]. We do not transform the

TABLE I: Descriptive statistics of the dataset.

Repository	Project Name (Abbreviation used in [2])	Key	#Issues	Story Point				
				Min	Max	Mean	Median	Std
Apache	Mesos (ME)	MESOS	1,680	1	40	3.09	3	2.42
	Usergrid (UG)	USERGRID	482	1	8	2.85	3	1.40
Appcelerator	Appcelerator Studio (AS)	TISTUD	2,919	1	40	5.64	5	3.33
	Aptana Studio (AP)	APSTUD	829	1	40	8.02	8	5.95
	Titanium (TI)	TIMOB	2,251	1	34	6.32	5	5.10
Atlassian	Bamboo (BB)	BAM	521	1	20	2.42	2	2.14
	Clover (CV)	CLOV	384	1	40	4.59	2	6.55
	Jira Server and Data Center (JI)	JRESERVER	352	1	20	4.43	3	3.51
DuraSpace	Duracloud (DC)	DURACLOUD	666	1	16	2.13	1	2.03
Lsstcorp	Data Management (DM)	DM	4,667	1	100	9.57	4	16.60
Moodle	Moodle (MD)	MDL	1,166	1	100	15.54	8	21.65
MuleSoft	Mule (MU)	MULE	889	1	21	4.90	5	3.61
	Mule Studio (MS)	MULESTUDIO	732	1	34	6.40	5	5.39
Spring	SpringXD (XD)	XD	3,526	1	40	3.70	3	3.23
Talendforge	Talend Data Quality (TD)	TDQ	1,381	1	40	5.92	5	5.19
	Talend ESB (TE)	TESB	868	1	13	2.16	2	1.50
Total			23,313					

$\hat{A}_{12}$ as we are interested in *any* improvement achieved by the methods [15], [18].

To perform the above analyses, we used the Wilcoxon Rank-Sum test and Vargha Delaney's $\hat{A}_{12}$ effect size available from the `stats` library in `R` v. 4.0.1 [19].

III. Results

RQ1. Does GPT2SP outperform Deep-SE and the baselines for the within-project scenario?

RQ1 aims at empirically assessing the prediction performance of GPT2SP and compare them with the predictions obtained by three other methods, namely Deep-SE, Mean Effort and Median Effort.

Table II shows the results we obtained by running GPT2SP, Deep-SE, and the Mean and Median baselines on each of the 16 software projects considered in our study. We can observe that, for the within-project scenario, GPT2SP outperforms the Median Effort baseline or Deep-SE in only six cases out of 16. Compared to the Mean Effort baseline, GPT2SP achieves a lower MAE in 14 cases, and a very similar one in the remaining two cases (i.e., Mule, Usergrid).

The results of the Wilcoxon Ranked-Sum test ($\alpha = 0.05$) applied to the distribution of absolute errors (see Table IV) reveal that GPT2SP provides statistically significant better estimates than using Median Effort in only three cases (two with negligible and one with small effect size), better estimates than Deep-SE only in two cases (both with negligible effect size). If we consider Bonferroni correction (by setting $\alpha = 0.016$ for $K = 3$ hypothesis) the number of statistically significant cases goes down to two and one for Median Effort and Deep-SE, respectively. While, the difference between the absolute errors obtained by GPT2SP and by Mean Effort is statistically significant in 12 out of 16 cases in favour of GPT2SP, even when considering the Bonferroni correction.

We conclude that the Median Effort baseline is the best predictor for the within-project scenario achieving the lowest MAE in 8 (50%) cases, followed by GPT2SP, which achieves the lowest MAE in 5 (31%) cases, and Deep-SE achieving the lowest MAE in only 3 (19%) cases.

Therefore, our replication disproves the results previously obtained by Fu and Tantithamthavorn [1], that is we found that GPT2SP does not outperform Deep-SE or the Median Effort baseline in all cases for the within-project scenario.

For completeness, we report the MAE values obtained by the original study [1] in Table III (which uses an incorrect formula to compute the MAE as we explained in Section II-D) as well as those obtained by our replication.[2] We can observe that the computation error of the MAE observed in the original study, led to incorrect values which can be up to three times lower (i.e., better) than the correctly computed value, depending on the project, thus inflating GPT2SP's results reported in the original study [1].

RQ2. Does GPT2SP outperform Deep-SE and the baselines for the cross-project scenario?

As done in the original study by Fu and Tantithamthavorn, we also investigated two cross-project estimation (CP) scenarios, namely the within-repository (CPWR) and cross-repository (CPCR) scenarios.

Table Va shows the results we achieved for GPT2SP, Deep-SE and the Mean Effort and Median Effort baselines for the CP within-repository scenario (CPWR). The results of the Wilcoxon test on the distribution of the absolute errors

[2]The authors did not report the MdAE values in their article or GitHub repository, as a result, we only compare the MAE values in Table III.

TABLE II: RQ1. Results obtained by GPT2SP, Deep-SE, and the two baselines (Mean and Median) for the Within-Project scenario. Best results (among all approaches per project) are highlighted in bold.

Project	Method	MAE	MdAE	SA	Project	Method	MAE	MdAE	SA
Mesos	GPT2SP	1.21	0.98	42.56	Duracloud	GPT2SP	**0.80**	**0.50**	**47.97**
	Deep-SE	**1.12**	**0.73**	**46.72**		Deep-SE	0.82	0.53	46.97
	Mean	1.41	1.78	33.18		Mean	1.00	1.14	35.17
	Median	1.22	2.00	42.30		Median	0.82	1.00	46.78
Usergrid	GPT2SP	1.19	1.02	22.56	Data Management	GPT2SP	**5.39**	**2.00**	**54.82**
	Deep-SE	1.18	**0.80**	23.25		Deep-SE	5.86	2.22	50.87
	Mean	1.19	1.23	22.43		Mean	8.66	4.55	27.35
	Median	**1.15**	1.00	**24.97**		Median	6.19	3.00	48.09
Appcelerator Studio	GPT2SP	1.53	0.71	49.88	Moodle	GPT2SP	8.38	8.53	45.76
	Deep-SE	1.42	**0.58**	53.49		Deep-SE	7.89	**4.93**	48.92
	Mean	1.91	1.52	37.52		Mean	12.63	12.11	18.21
	Median	**1.30**	1.00	**57.41**		Median	**6.59**	6.00	**57.32**
Aptana Studio	GPT2SP	**3.52**	3.10	**37.75**	Mule	GPT2SP	2.61	2.23	28.55
	Deep-SE	4.14	**2.52**	26.74		Deep-SE	2.59	**1.96**	29.14
	Mean	3.59	3.46	36.53		Mean	2.60	2.22	28.72
	Median	3.61	4.00	36.18		Median	**2.47**	2.00	**32.29**
Titanium	GPT2SP	2.35	1.45	48.78	Mulesoft	GPT2SP	3.70	2.62	23.55
	Deep-SE	2.09	**1.34**	54.49		Deep-SE	3.67	**2.26**	24.12
	Mean	3.02	1.97	34.29		Mean	3.74	2.80	22.65
	Median	**2.04**	2.00	**55.71**		Median	**3.66**	3.00	**24.32**
Bamboo	GPT2SP	0.77	0.69	54.30	Spring XD	GPT2SP	1.78	1.54	37.09
	Deep-SE	0.81	**0.61**	51.78		Deep-SE	**1.70**	**1.31**	**39.96**
	Mean	1.22	1.31	27.99		Mean	2.05	2.53	27.59
	Median	**0.75**	1.00	**55.34**		Median	1.71	2.00	39.55
Clover	GPT2SP	3.76	0.97	32.89	Talend Data Quality	GPT2SP	3.65	3.44	29.41
	Deep-SE	**3.39**	**0.80**	**39.41**		Deep-SE	3.61	**2.92**	30.07
	Mean	4.57	3.06	18.46		Mean	4.56	5.08	11.69
	Median	3.71	2.00	33.77		Median	**3.31**	4.00	**35.85**
Jira Software	GPT2SP	**1.57**	**0.81**	**54.09**	Talend ESB	GPT2SP	**0.86**	**0.55**	**38.81**
	Deep-SE	1.70	1.09	50.08		Deep-SE	0.90	0.59	36.19
	Mean	2.40	2.15	29.61		Mean	1.04	0.91	26.47
	Median	2.31	2.00	32.40		Median	0.92	1.00	34.86

produced by GPT2SP against each of the other methods are provided in the last column of this table.

We can observe that GPT2SP outperforms Deep-SE in only two out of eight cases (with marginal statistical significance and negligible effect size in only one of the two cases), whereas Deep-SE outperforms GPT2SP in six cases.

Moreover, GPT2SP outperforms the Median Effort baseline in only two cases out of eight, with statistical significance but negligible effect size in both cases. GPT2SP achieves a lower MAE than that of the Mean Effort baseline in five cases, a higher one in two, and an equal one in the remaining one case out of eight cases, where the difference is significant for three cases but always with a negligible effect size.

When considering the results of the cross-repository (CPCR) scenario (see Table Vb), we observe that GPT2SP outperforms Deep-SE in six cases out of eight with a statistically significant difference in five of them out of which one shows a medium effect size, one a small one, and three a negligible one. GPT2SP achieves a lower MAE than that obtained by Median Effort in three cases only, among which one case shows a small effect size and two other cases show a negligible effect size. GPT2SP outperforms the Mean Effort baseline in seven cases with a statistically significant difference in six with a large effect size in three, medium one in two and small in one.

Overall, considering both cross-project scenarios, we observed that overall GPT2SP performs better than Deep-SE, which is the same conclusion achieved by the original study. However, GPT2SP performs poorly when compared against the Median Effort baseline, that is, it achieves statistically significantly better results in less than a third of the cases (i.e., five out of 16 cases). Therefore, our replication does not support the conclusion made in the original study stating that GPT2SP outperforms the baselines statistically significantly for all cases in the cross-project scenarios.

IV. Threats to Validity

Threats to construct validity relate to the quality of story point datasets. Prior studies on defect and vulnerability prediction raised concerns that the prediction models are inaccurate if the ground-truth labels are noisy [20], [21]. Noisy labels are generally caused by the use of heuristics to generate ground-truth data.

TABLE III: RQ1. The MAE values obtained by the original study (GPT2SP$_{orig}$) Vs. those obtained by this replication (GPT2SP$_{rep}$) for within-project estimation.

Project	GPT2SP$_{orig}$	GPT2SP$_{rep}$
Mesos	0.66	1.21
Usergrid	0.68	1.19
Appcelerator Studio	0.84	1.53
Aptana Studio	1.93	3.52
Titanium	1.36	2.35
Bamboo	0.44	0.77
Clover	1.98	3.76
Jira Software	0.92	1.57
Duracloud	0.48	0.80
Data Management	3.10	5.39
Moodle	4.09	8.38
Mule	1.43	2.61
Mulesoft	2.04	3.70
Spring XD	0.96	1.78
Talend Data Quality	1.58	3.65
Talend ESB	0.50	0.86

TABLE IV: RQ1. Results of the Wilcoxon test (with Vargha-Delaney $\hat{A}_{12}$ effect size in parentheses) comparing the absolute errors of GPT2SP to that of Deep-SE and the two baselines (i.e., Mean and Median).

Project	GPT2SP vs		
	Mean	Median	Deep-SE
Mesos	0.127 (0.55) _	0.955 (0.43) _	0.248 (0.53) _
Usergrid	<0.001 (0.70) M	1.000 (0.43) _	0.999 (0.44) _
Appcelerator Studio	<0.001 (0.64) S	1.000 (0.37) _	1.000 (0.43) _
Aptana Studio	0.713 (0.48) _	0.679 (0.49) _	0.044 (0.55) N
Titanium	<0.001 (0.70) M	0.583 (0.49) _	0.516 (0.50) _
Bamboo	<0.001 (0.70) M	0.021 (0.59) N	0.412 (0.51) _
Clover	<0.001 (0.76) M	0.004 (0.54) N	0.001 (0.54) N
Jira Software	<0.001 (0.62) S	0.988 (0.42) _	0.478 (0.50) _
Duracloud	<0.001 (0.58) N	0.663 (0.49) _	0.972 (0.46) _
Data Management	<0.001 (0.79) M	1.000 (0.31) _	1.000 (0.40) _
Moodle	0.323 (0.51) _	0.678 (0.49) _	0.294 (0.52) _
Mule	0.497 (0.50) _	0.751 (0.48) _	0.722 (0.48) _
Mulesoft	<0.001 (0.59) N	0.962 (0.47) _	0.843 (0.48) _
Spring XD	<0.001 (0.66) S	0.977 (0.45) _	0.433 (0.50) _
Talend Data Quality	<0.001 (0.61) S	0.130 (0.53) _	0.325 (0.51) _
Talend ESB	<0.001 (0.68) S	<0.001 (0.65) S	0.217 (0.54) _

The dataset used herein as well as the original study was curated by Choetkiertikul et al. [2], and used herein, collect story point data based on actual information provided in the JIRA issue tracking system. This mitigates the risk of inaccurate labelling, however, this data is input into the system by engineers and thus it can be subject to human errors. Overall, the quality of the data should not pose a fundamental threat to our replication.

Threats to internal validity can arise from the hyperparameter settings of GPT2SP. In our replication, we use the same settings used for GPT2SP in the original study (as reported in their replication package), and as such we inherit the same threats. In fact, the GPT2SP setting consists of various hyperparameters (i.e., number of hidden layers, number of attention heads, and learning rate) and finding an optimal setting is very expensive given the large search space of the

TABLE V: RQ2. Comparing GPT2SP performance with Deep-SE and the baselines in two cross-project scenarios: (a) Within- and (b) Cross-Repository. The results of the Wilcoxon test ($\hat{A}_{12}$ effect size in parentheses) for GPT2SP vs. Deep-SE and baselines are shown in the last column. Best results (among all approaches per project) are highlighted in bold.

(a) Within-Repository

Source	Target	Method	MAE	MdAE	SA	GPT2SP vs.
MESOS (ME)	USERGRID (UG)	GPT2SP	1.11	0.84	42.49	
		Deep-SE	1.16	0.96	39.84	0.049 (0.53) N
		Mean	1.02	0.19	46.99	0.998 (0.45) _
		Median	**0.89**	**0.00**	**54.10**	1.000 (0.37) _
USERGRID (UG)	MESOS (ME)	GPT2SP	1.52	0.92	15.64	
		Deep-SE	1.51	1.01	16.18	0.662 (0.50) _
		Mean	1.52	**0.80**	15.57	0.325 (0.50) _
		Median	**1.50**	1.00	**16.27**	0.692 (0.50) _
TISTUD (AS)	APSTUD (AP)	GPT2SP	4.62	3.37	7.99	
		Deep-SE	4.37	2.98	12.99	1.000 (0.45) _
		Mean	**4.27**	**2.18**	**15.05**	0.992 (0.47) _
		Median	4.38	3.00	12.73	1.000 (0.43) _
TISTUD (AS)	TIMOB (TI)	GPT2SP	3.28	2.47	22.60	
		Deep-SE	3.38	2.39	20.31	0.253 (0.51) _
		Mean	3.45	2.82	18.69	<0.001 (0.56) N
		Median	**3.17**	**2.00**	**25.24**	1.000 (0.44) _
APSTUD (AP)	TISTUD (AS)	GPT2SP	2.86	2.45	45.15	
		Deep-SE	**2.70**	**2.07**	**48.25**	1.000 (0.47) _
		Mean	3.38	3.24	35.20	<0.001 (0.58) N
		Median	3.17	3.00	39.30	<0.001 (0.55) N
APSTUD (AP)	TIMOB (TI)	GPT2SP	4.09	3.52	30.38	
		Deep-SE	**3.51**	**2.53**	**40.34**	1.000 (0.39) _
		Mean	4.36	4.24	25.78	<0.001 (0.56) N
		Median	4.19	4.00	28.67	0.004 (0.52) N
MULE (MU)	MULESTUDIO (MS)	GPT2SP	3.48	2.29	21.32	
		Deep-SE	3.64	**2.04**	17.61	0.206 (0.51) _
		Mean	3.34	2.71	24.42	0.943 (0.48) _
		Median	**3.26**	3.00	**26.26**	0.999 (0.45) _
MULESTUDIO (MS)	MULE (MU)	GPT2SP	3.03	2.65	30.27	
		Deep-SE	2.77	2.47	36.28	0.985 (0.47) _
		Mean	3.05	**1.77**	29.83	0.078 (0.52) _
		Median	**2.60**	3.00	**40.24**	1.000 (0.43) _

(b) Cross-Repository

Source	Target	Method	MAE	MdAE	SA	GPT2SP vs.
TISTUD (AS)	USERGRID (UG)	GPT2SP	**2.18**	2.03	**37.67**	
		Deep-SE	3.47	3.50	0.98	<0.001 (0.76) M
		Mean	3.08	2.82	12.02	<0.001 (0.73) M
		Median	2.30	**2.00**	34.40	<0.001 (0.56) N
TISTUD (AS)	MESOS (ME)	GPT2SP	2.65	**2.75**	28.83	
		Deep-SE	3.18	3.20	14.70	<0.001 (0.60) S
		Mean	3.28	2.82	11.95	<0.001 (0.64) S
		Median	**2.58**	3.00	**30.85**	0.994 (0.47) _
MDL (MD)	APSTUD (AP)	GPT2SP	4.37	3.07	68.12	
		Deep-SE	5.03	3.77	63.29	<0.001 (0.55) N
		Mean	9.84	8.95	28.21	<0.001 (0.86) L
		Median	**3.97**	**3.00**	**71.05**	0.989 (0.47) _
MDL (MD)	TIMOB (TI)	GPT2SP	3.60	2.53	73.90	
		Deep-SE	**3.34**	**1.96**	**75.82**	1.000 (0.45) _
		Mean	11.19	11.95	18.91	<0.001 (0.91) L
		Median	4.19	4.00	69.63	<0.001 (0.60) S
MDL (MD)	TISTUD (AS)	GPT2SP	**2.16**	1.96	**83.92**	
		Deep-SE	2.64	**1.66**	80.35	<0.001 (0.57) N
		Mean	11.45	11.95	14.90	<0.001 (0.98) L
		Median	3.17	3.00	76.44	<0.001 (0.61) S
DM	TIMOB (TI)	GPT2SP	3.58	2.31	61.99	
		Deep-SE	3.81	2.65	59.52	<0.001 (0.53) N
		Mean	5.61	5.03	40.35	<0.001 (0.75) M
		Median	**3.46**	**1.00**	**63.22**	0.999 (0.47) _
USERGRID (UG)	MULESTUDIO (MS)	GPT2SP	4.02	2.16	4.57	
		Deep-SE	3.95	2.11	6.18	0.865 (0.48) _
		Mean	4.04	2.20	4.00	0.145 (0.52) _
		Median	**3.91**	**2.00**	**7.16**	0.996 (0.46) _
MESOS (ME)	MULE (MU)	GPT2SP	3.13	2.24	9.57	
		Deep-SE	3.20	2.31	7.37	0.347 (0.51) _
		Mean	**2.89**	**1.81**	**16.56**	0.999 (0.46) _
		Median	2.92	2.00	15.65	0.999 (0.46) _

Transformer architecture. Given the goal of the original study was not to find the best hyperparameter setting, they did not perform any tuning and stated that the accuracy reported in their paper should serve as a lower bound of the performance of GPT2SP, which may be improved through hyperparameter optimization.

To straighten *conclusion validity*, we carefully computed the evaluation measures and applied statistical tests by verifying all the required assumptions.

Threats to the external validity relate to the generalizability of the accuracy of our GPT2SP approach. The results obtained herein are based on 23,313 issues from open-source projects, future work can enhance the external validity by exploring different open-source (e.g., TAWOS [22]) or proprietary software projects.

V. Related Work

Effort estimation in Agile software development poses unique challenges due to the dynamic and iterative nature of Agile projects. Recent research has explored the application of deep learning techniques to address these challenges and enhance the accuracy of effort estimation in Agile contexts. This section provides an overview of key studies in the application of deep learning for Agile software effort estimation.

The first study using Neural Networks (NN) for Agile effort estimation was published in 2011 by Abrahamsson et al. [23]. They trained a model on 17 features extracted from user stories such as the priority and number of the characters in the user story, and 15 binary variables representing the occurrence of 15 keywords in the user stories. They applied their model to two industrial case studies, one consisting of 1,325 user stories and the other of 13 user stories. The models were able to predict story points with a relative root square error of 44% for one case study, but more than 100% for the other.

Soares [24] used an NLP technique based on auto-encoder neural networks to classify user stories based on the semantic differences in their title into story point classes. They used TF/IDF and document embedding with four variants of auto-encoders, and evaluated these models on 3,439 issue reports from six open-source projects. The results revealed no significant difference in the SP estimation accuracy of these approaches. Soares speculated that this might be due to the relative semantic simplicity of issue report titles.

Choetkiertikul et al. [2] proposed Deep-SE, a new approach based on the combination of two deep learning architectures to build an end-to-end prediction system for SP estimation. They used raw user story text as input to their system. The model generates word embeddings based on pre-training on previous user stories and maps the embeddings to the target SP value using a regression model. They evaluated Deep-SE on 23,313 issues from 16 open-source projects and showed that it outperforms benchmarks which included Mean and Median baselines. Subsequently, Abadeer and Sabetzadeh [25] evaluated the effectiveness of Deep-SE for SP prediction with a commercial dataset of 4,727 user stories collected from a healthcare data science company. They found that Deep-SE outperforms random guessing, Mean and Median baselines statistically significantly, however with a small effect size.

Tawosi et al. [3] replicated the Deep-SE [2] study and extended the evaluation to a larger and more diverse dataset. After fixing a bug in Deep-SE's evaluation code, they found that Deep-SE outperformed the baseline methods statistically significantly in only 46% of the cases which is less than half the cases.

In a recent study, Tawosi et al. [26] used a Large Language Model (LLM) (GPT4, specifically) to estimate story points for Agile user stories from three open-source projects. The results showed that the LLM's estimation performance was inferior to that of the Mean and Median baselines in a zero-shot setting. Thus, the authors used Multi-Objective Optimisation techniques to find the optimal set of example user stories to improve the LLM's estimation performance in a few-shot setting. The final results improved the LLM's estimation performance to outperform both the baselines in all three projects.

In summary, the application of deep learning for agile software effort estimation presents a dynamic and evolving field. Researchers continue to explore novel architectures, representation learning techniques, and real-world applications to advance the capabilities of deep learning models in predicting software development effort. Despite advancements, challenges remain, including model accuracy, interpretability, handling diverse data sources, and addressing uncertainties.

VI. Conclusions

In this paper, we presented a replication of the study by Fu and Tantithamthavorn [1] published in the IEEE Transactions on Software Engineering in 2022. The authors proposed GPT2SP, a Transformer-based deep learning model for Story Point (SP) estimation of user stories with the aim of overcoming the limitations of Deep-SE, a method previously proposed by Choetkiertikul et al. [2].

Our replication effort allowed us to reveal a bug in the source code used in the original work [1], which has caused a discrepancy in the results reported in the original study and obtained herein by using a bug-free version of the same code. Such a discrepancy is not negligible, in fact, it changed the main finding of the original work: We showed that the proposed approach is not more effective than a mere Median Effort baseline. Based on these results, GPT2SP cannot be considered the current state-of-the-art approach to agile effort development estimation. On the other hand, Deep-SE was also shown to be not effective in a previous replication study [3]. Taken together, the results of these two replication studies, suggest that *there is no effective state-of-the-art approach for agile effort estimation, and further research is needed in this area* to find suitable techniques that can provide engineers with accurate agile software development effort estimates. Moreover, based on our experience undertaking replication studies, we would like to highlight two more general take-aways. We understand that researchers can sometimes unintentionally make mistakes in their studies, however these *mistakes can*

be revealed and fixed by subsequent research when a study provides enough information/material to be replicated, as in the case of the work by Fu and Tantithamthavorn [1] and Choetkiertikul et al. [2], who made publicly available their data and source code. Moreover, replication studies, as well as negative results, are important and should be encouraged and supported by the software engineering research community. Researchers should work together towards this end. We contacted the authors of the original study to share the results of our replication before submitting our work for peer-review, in order to check with them whether our understanding of their work and our findings were sound. The authors were responsive and supportive and even incorporated our suggested fix in their repository. We believe that such collegial behaviour in support of replication studies is exemplary and should be embraced by our community in order to strengthen the foundation of Empirical Software Engineering Research.

Data Availability

The data used in this study is publicly available at [12], while the correct source code for GPT2SP and Deep-SE is available at https://github.com/awsm-research/gpt2sp/pull/2 and https://github.com/SOLAR-group/AgileEffortEstimation, respectively.

Acknowledgments

This work has been supported by the European Research Council Advanced Grant no.741278.

We would like to thank the authors of the original study for their responsiveness and openness in discussing their work with us, thereby supporting our replication study.

References

[1] M. Fu and C. Tantithamthavorn, "GPT2SP: A transformer-based agile story point estimation approach," *IEEE Transactions on Software Engineering*, 2022.

[2] M. Choetkiertikul, H. K. Dam, T. Tran, T. Pham, A. Ghose, and T. Menzies, "A deep learning model for estimating story points," *IEEE Transactions on Software Engineering*, vol. 45, no. 7, pp. 637–656, 2019.

[3] V. Tawosi, R. Moussa, and F. Sarro, "Agile effort estimation: Have we solved the problem yet? insights from a replication study," *IEEE Transactions on Software Engineering*, 2022.

[4] A. Radford, K. Narasimhan, T. Salimans, I. Sutskever *et al.*, "Improving language understanding by generative pre-training," 2018.

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

[6] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[7] F. A. Gers, J. Schmidhuber, and F. Cummins, "Learning to forget: Continual prediction with LSTM," *Neural Computation*, 1999.

[8] T. Pham, T. Tran, D. Phung, and S. Venkatesh, "Faster training of very deep networks via p-norm gates," in *2016 23rd International Conference on Pattern Recognition (ICPR)*. IEEE, 2016, pp. 3542–3547.

[9] N. Mittas, I. Mamalikidis, and L. Angelis, "A framework for comparing multiple cost estimation methods using an automated visualization toolkit," *Information and Software Technology*, vol. 57, pp. 310–328, 2015.

[10] P. A. Whigham, C. A. Owen, and S. G. Macdonell, "A baseline model for software effort estimation," *ACM Transactions on Software Engineering and Methodology (TOSEM)*, vol. 24, no. 3, pp. 1–11, 2015.

[11] V. Tawosi, F. Sarro, A. Petrozziello, and M. Harman, "Multi-objective software effort estimation: a replication study," *IEEE Transactions on Software Engineering (TSE)*, vol. 48, no. 8, pp. 3185–3205, 2021.

[12] "Deep-SE Source Code GitHub." [Online]. Available: https://github.com/morakotch/datasets/tree/master/storypoint/IEEE\%20TSE2018

[13] M. Shepperd and S. MacDonell, "Evaluating prediction systems in software project estimation," *Information and Software Technology*, vol. 54, no. 8, pp. 820–827, 2012.

[14] W. B. Langdon, J. Dolado, F. Sarro, and M. Harman, "Exact mean absolute error of baseline predictor, MARP0," *Information and Software Technology*, vol. 73, pp. 16–18, 2016.

[15] F. Sarro, A. Petrozziello, and M. Harman, "Multi-objective software effort estimation," in *2016 IEEE/ACM 38th International Conference on Software Engineering (ICSE)*. IEEE, 2016, pp. 619–630.

[16] F. Sarro, R. Moussa, A. Petrozziello, and M. Harman, "Learning from mistakes: Machine learning enhanced human expert effort estimates," *IEEE Transactions on Software Engineering*, 2020.

[17] A. Arcuri and L. Briand, "A hitchhiker's guide to statistical tests for assessing randomized algorithms in software engineering," *Software Testing, Verification and Reliability*, vol. 24, no. 3, pp. 219–250, 2014.

[18] G. Neumann, M. Harman, and S. Poulding, "Transformed vargha-delaney effect size," in *International Symposium on Search Based Software Engineering*. Springer, 2015, pp. 318–324.

[19] "R: The R Project for Statistical Computing, V. 4.0.1," 2020. [Online]. Available: https://www.r-project.org/

[20] 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 *2015 IEEE/ACM 37th IEEE International Conference on Software Engineering*, vol. 1, 2015, pp. 812–823.

[21] M. Jimenez, R. Rwemalika, M. Papadakis, F. Sarro, Y. Le Traon, and M. Harman, "The importance of accounting for real-world labelling when predicting software vulnerabilities," in *Proceedings of the 2019 27th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, ser. ESEC/FSE 2019. New York, NY, USA: Association for Computing Machinery, 2019, p. 695–705. [Online]. Available: https://doi.org/10.1145/3338906.3338941

[22] V. Tawosi, A. Al-Subaihin, R. Moussa, and F. Sarro, "A versatile dataset of agile open source software projects," in *Proceedings of the 19th International Conference on Mining Software Repositories*, ser. MSR '22. New York, NY, USA: Association for Computing Machinery, 2022, p. 707–711. [Online]. Available: https://doi.org/10.1145/3524842.3528029

[23] P. Abrahamsson, I. Fronza, R. Moser, J. Vlasenko, and W. Pedrycz, "Predicting development effort from user stories," in *ESEM*. IEEE, 2011, pp. 400–403.

[24] R. G. Soares, "Effort estimation via text classification and autoencoders," in *2018 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2018, pp. 01–08.

[25] M. Abadeer and M. Sabetzadeh, "Machine learning-based estimation of story points in agile development: Industrial experience and lessons learned," in *REW*. IEEE, 2021, pp. 106–115.

[26] V. Tawosi, S. Alamir, and X. Liu, "Search-based optimisation of LLM learning shots for story point estimation," in *15th International Symposium on Search-Based Software Engineering*. Springer, 2023.