Investigating the Effectiveness of Clustering for Story Point Estimation

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

Afnan Al-Subaihin
Department of Computer Science
University College London (UCL)
London, United Kingdom
a.alsubaihin@ucl.ac.uk

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

***Abstract*—Automated techniques to estimate Story Points (SP) for user stories in agile software development came to the fore a decade ago. Yet, the state-of-the-art estimation techniques' accuracy has room for improvement.**

In this paper, we present a new approach for SP estimation, based on analysing textual features of software issues by employing latent Dirichlet allocation (LDA) and clustering. We first use LDA to represent issue reports in a new space of generated topics. We then use hierarchical clustering to agglomerate issues into clusters based on their topic similarities. Next, we build estimation models using the issues in each cluster. Then, we find the closest cluster to the new coming issue and use the model from that cluster to estimate the SP.

Our approach is evaluated on a dataset of 26 open source projects with a total of 31,960 issues and compared against both baselines and state-of-the-art SP estimation techniques.

The results show that the estimation performance of our proposed approach is as good as the state-of-the-art. However, none of these approaches is statistically significantly better than more naive estimators in all cases, which does not justify their additional complexity. We therefore encourage future work to develop alternative strategies for story points estimation.

The experimental data and scripts we used in this work are publicly available to allow for replication and extension.

***Index Terms*—Software Effort Estimation, Story Point Estimation, Latent Dirichlet Allocation, Hierarchical Clustering**

I. Introduction

In agile development, Story Point (SP) is a commonly used measure of the complexity and required effort of completing a software development task, [1], [2]. Teams typically carry out assigning story points to these tasks in order to plan for the content of upcoming sprints. To this end, teams mainly rely on expert estimation methods like Planning Poker and Delphi [3]. However, expert judgment has been shown to be prone to bias due to its reliance on subjective assessment [4]–[6]. This motivated several research endeavours to find automated ways to predict story points of a task given its features with the aim of avoiding inaccurate estimations by human judgement, in addition to, and more importantly to an agile team, producing consistent estimations throughout the project's lifecycle.

Task descriptions (referred to as *user stories* in agile development) are a convenient information source for both humans and automated SP estimators. This information, which is usually conveyed via few sentences written in natural language by product owners, developers, or users, is available upon the creation of a new task. Most SP estimation techniques study the similarities between the task at hand and the previously completed tasks to decide on the SP value of the new task [7], [10]–[14].

On the other hand, previous studies in software effort estimation showed that software engineering data often contain a large amount of variability [23], [27]; as previously shown in the literature for traditional software effort estimation [18]–[21], [23], [24], [26].

Therefore, in order to help reduce such variability, we propose and investigate the suitability of a novel clustering-based model to estimate the SP values of new issues. We dub our proposed approach **L**DA-based **H**ierarchical **C**lustering for **S**tory point **E**stimation (**LHC-SE**), hereafter.

The proposed approach, LHC-SE, relies on the similarities of historical issue descriptions by grouping them into coherent clusters, that maximize their prediction power. These clusters are formed by representing issues as vectors of their LDA-extracted topics; the estimation of SP for a new issue is then inferred from the SP scores of issues in its assigned cluster. We report the results of three SP assignments (estimation models) based on the resulting clusters: first is assigning the issue at hand the mean SP of the issues in its assigned cluster, the median of the aforementioned, and finally, assigning it the SP of the issue deemed most similar.

In order to evaluate the proposed approach, we use the largest publicly available dataset of issues mined from 26 agile projects to date [44]. The SP values estimated by human experts are used as the ground truth to evaluate the accuracy of our estimation model. Accuracy is measured by using robust measures such as the mean and median absolute errors of the estimations, and the relative improvement over random guessing [30].

The results show that clustering issues based on their topic similarity (i.e., LHC-SE) improves the accuracy of estimation over Random Guessing and Mean baseline method, with statistical significance. While it is comparable to the state-of-the-art SP methods proposed in literature. We also observed that the Median baseline estimator achieves similar accuracy as our model and the state-of-the-art on this dataset. We discuss these results and our observations in details in Section VI.

II. Related Work

Initial research proposing a technique to predict the Story Point (SP) of an issue based on its description appeared almost a decade ago. These studies focused mostly on producing models that act only as a decision support system for expert estimators in agile teams. Abrahamsson et al. [10] were the first to propose an automated method for SP estimation. They used 17 features extracted from the user story, including priority, the number of the characters in the user story and 15 binary variables presenting the occurrence of 15 keywords in the user story, to train Machine Learning (ML) methods. They evaluated their approach on an industrial dataset of 1,338 issues and found this feature set effective to achieve a good accuracy.

Porru et al. [11] treated the SP estimation problem as a classification problem. Their approach used features extracted from 4,908 issue descriptions collected from open-source repositories. Specifically, they extract the type of the issue, the component(s) related to it, and the Term Frequency-Inverse Document Frequency (TF-IDF) derived from the title and description of the issue. They confirmed Abrahamsson et alia's findings that user stories and their length are useful predictors for story point estimation. Their results also indicate that more than 200 issues were needed for training a classifier that can serve as a stable model with satisfactory accuracy.

Scott and Pfahl [12] used developer-related features alongside the features extracted from 4,142 user stories collected from eight open-source projects to estimate the story points using ML tools. Developer-related features include the developer's reputation, workload, work capacity, and the number of comments. The results showed that using only developers' features as the input for the estimation model outperforms random guessing, mean, and median baseline estimators. This model also outperformed two other models: one that uses only features extracted from the text and another one using both the developers' features and text features together.

Soares [13] used auto-encoder neural networks to classify user stories based on their semantic differences in order to estimate their SP. The study used four variants of auto-encoders and found no significant difference between them with respect to their SP estimation accuracy. Soares speculated that the relative semantic simplicity of user stories lead to these results. He evaluated these methods using 3,439 issue reports collected from six open source project.

Choetkiertikul et al. [14] combined two deep learning architectures to build an end-to-end SP prediction system, which they call Deep-SE. The input to their system is raw user story text. Through deep-learning, they convert user stories into a fixed-length vector, which is then fed to a regressor that maps the deep representation to the output SP estimation. Deep-SE is evaluated using 23,313 issues collected from nine open-source repositories. They found that Deep-SE outperforms the approach proposed by Porru et al. [11] and baseline benchmarks in terms of MAE, statistically significantly. Recently, Abadeer and Sabetzadeh [39] adopted Deep-SE and evaluated it on a commercial project with 4,727 user stories. They found Deep-SE applicable in a non-open source setting as it outperformed the baselines statistically significantly.

To the best of our knowledge, our study is the first to investigate whether clustering can help improve SP estimation accuracy by reducing the variance in the issue descriptors. This is motivated by the observations made in previous work on traditional software effort estimation where clustering techniques are employed to reduce the variability, which often lead to construction of more accurate effort estimation models [18], [19], [21], [23], [24], [26], [42], [43]. Previous work that use clustering mainly use it to group together projects according to attributes that are pertinent to the task of effort estimation (i.e., cost drivers) such as size measures or manager/team experience. Whereas our study uses clustering to group the issue reports according to their topic. This makes the approach independent from the cost-drivers used as the basis for the estimation. Furthermore, our study employs the largest dataset (26 open-source projects and a total of 31,960 issues) used for SP estimation thus far [44].

III. The Proposed Method

The proposed method relies on clustering similarly described issues, such that, given an issue with an unknown SP score, its score can be derived from the SP scores of the issues deemed most similar to it (i.e. issues that reside in the same cluster).

In order to carry out the clustering, the similarities among issues need to be measured based on their natural language description. To this end, issues are represented as vectors in a numerical vector space, such that the distance among issues could be used as a proxy for issue similarity. In this study, we use topic modelling, namely, Latent Dirichlet Allocation (LDA); which uses statistical models to infer a set of topics in textual documents, and represents the documents as the set of probabilities of their relevance to each of the inferred topics [9], [28].

Equipped with a numerical representation of issues in a vector space, a clustering algorithm can be employed to group similar issues together. Many clustering techniques require the choice of k (number of clusters) to be known a priori, which is usually unknown for software engineering data [18]. In order to enable the discovery of a suitable k, we elect to use hierarchical clustering which produces a dendrogram that can be efficiently investigated for the most suitable cut-off point [8]. In this study, this choice of k is guided by estimation accuracy on a validation subset of the dataset. Using the resulting clusters, when a new issue needs to be assigned an SP score, it is first converted to a vector using the pre-existing LDA model. Afterwards, the most suitable cluster for the issue is identified as the one to which the closest issue belongs. Once the cluster of most similar issues is identified, we report the results of first assigning the issue the mean SP scores of issues in the similar cluster, the median, and finally, simply assigning the issue the SP score of the most similar issue in the historical dataset.

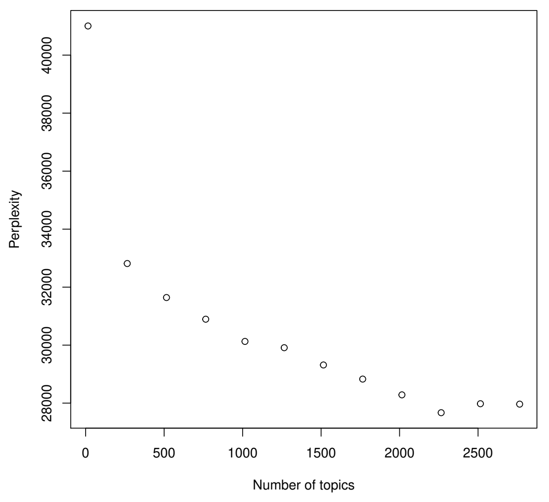

Fig. 1. Perplexity of the LDA topic model per number of topics (i.e., t-values).

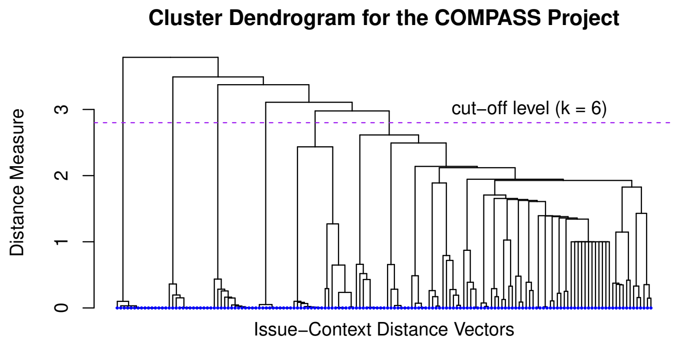

Fig. 2. A sample dendrogram of agglomerative hierarchical clustering of issues (COMPASS project). A sample cut-off line is shown on the plot, which cuts the dendrogram at level 6, thus producing 6 clusters.

A. Text Pre-processing and Topic Modelling

To capture the context of issue reports and their purpose, the title and the description of the issue are combined, dubbed the issue-context hereafter. In order to create a vector representation of the issue-context, we first perform basic text cleaning and pre-processing operations on the text. Specifically, we remove URLs, code snippets, and all non-alphanumeric characters from the issue-context, convert the text to lowercase, remove punctuation, remove English stop-words, and remove words with less than two characters. We do not perform stemming, as previous work showed it is prone to over-stemming, which lowers the accuracy of the results [15].

To generate an LDA topic model, we first join all the pre-processed training issue-contexts from a designated training subset of issues in all 26 projects in the Tawosi dataset to build a large corpus. The training corpus was then fed into the LDA topic modelling algorithm (using the `topicmodels` library in `R`). In order to set the number of topics t needed by the generative model, we explore over the range of t values, to find the t that produces a model with the minimum perplexity. The perplexity was evaluated using the validation subset of the dataset which was not included in the initial model generation. The used LDA technique employs Gibbs sampling [29] to identify topics in the corpus, the α and δ parameters were set to $1/t$ and 0.1 respectively. Fig. 1 shows the perplexity of the models built for different t values. We find that the model with the least perplexity is produced with 2,265 topics. The generated topic model is then used to generate posterior probabilities for issues in the testing subset of the dataset; thus representing each issue as the vector of all topics such that each cell in the vector represents the relatedness of the issue to the topic.

B. Clustering

Given the generated vector space of the issues in the dataset, a clustering algorithm is used to cluster similar issues, with regards to their topics, to cohesive clusters. This is done using agglomerative hierarchical clustering (using Ward's linkage criterion [16] and cosine as a distance measure). This generates a dendrogram of the clustering options for each k. This dendrogram can be explored at various cut-off points. A sample dendrogram for the COMPASS project is shown in Fig. 2.

In order to discover the most suitable k, we perform a simple greedy search to find the cut-off point that generates the clustering solution that would produce the most accurate estimation models.

We investigate three strategies for selecting the most suitable cut-off k: (a) a k which when used, the estimation models of the resulting clustering produce the lowest Mean Absolute Error (i.e., *MAE-based* strategy), (b) a k which when used, the estimation models produce the lowest Median Absolute Error (i.e., *MdAE-based* strategy), and (c) a k which when used, the resulting clusters have the highest silhouette index (i.e., *Silhouette-based* strategy). The silhouette index is an internal measure of cluster quality by calculating how similar are the issues to each other in their own cluster (cohesion) compared to other clusters (separation) [17]. To evaluate the first two strategies, MAE and MdAE are calculated over the validation subset of the dataset. We examine a range of different k values per project, starting from 3 to $0.9 \times l$ with $\frac{l}{10}$ increments, where l is the size of the training set. We chose one k value per project using each of the three strategies, and perform experiments with all three strategies.

C. Estimation Models

Once a clustering solution is selected, an SP estimation model is built using issues in each cluster. We investigate three models: (a) *Cluster Mean-based* and (b) *Cluster Median-based* estimators, which return the mean/median SP of all the issues in the cluster, respectively; and (c) *Closest Point-based* estimator, which returns the SP value of the closest issue to the queried issue, as the estimated SP.

Given a new issue, the previously generated topic model is used to compute the posterior probabilities of the issue-context. Then, its *cosine* distance from all the issues used in the training phase is computed to determine the closest cluster to the issue at hand. Then, the estimation model of that cluster is used to estimate SP value for the new issue.

IV. Empirical Study Design

In order to evaluate the performance of LHC-SE, we investigate the accuracy of the SP estimation models it produces. Additionally, we explore whether additional features help improve the estimation accuracy of the clustering approach. Finally, we compare the resulting performance with other state-of-the-are techniques in SP effort estimation using natural language. To answer these question, we train, evaluate and validate using a dataset of issues collected from 26 projects, from 13 different open-source repositories. Following is a detailed report of the empirical study design.

A. Research Questions

We investigate three research questions to asses the effectiveness of LHC-SE, against the baseline methods and previous work.

1) RQ1. Does clustering of issue reports based on their textual similarities help accurately estimate story points?: To see whether the proposed approach is a suitable method for estimating story points, we compare it against baseline estimators. Specifically, we compare LHC-SE to Random guessing, Mean and Median baselines. Mean and Median baselines are simple models used for sanity checks; they mainly involve assigning the issue at hand the mean and median SP over all previous issues, respectively.

To be accepted as a suitable estimation method, LHC-SE should be able to outperform these baseline techniques.

2) RQ2. Can additional features help improve estimation accuracy of the clustering approach?: In a further investigation, we augment the LDA-generated topic probabilities extracted from issue-context with additional features in the vector space. In particular, we add two features that are available when the issue is created: the issue type (e.g. *story*, *improvement*, *bug*, etc.) and the component(s) from which the issue rose (e.g. *UI*, *Runtime*, *DSL*, etc.). Since type and component are categorical variables, we use a one-hot encoding to convert them to numerical features to be able to exploit them with our approach. We also add issue report length, which is the number of characters used to describe the issue. This can serve as an indicator of the complexity of the issue. Adding these features creates a new variant of our model, we call it LHC$_{TC}$-SE to distinguish it from the base LHC-SE model.

Furthermore, we add TF-IDF features to LHC$_{TC}$-SE features to see if they help the clustering approach achieve more coherent clusters, thus improve its estimation accuracy. We refer to this variant as LHC$_{TC+TFIDF}$-SE.

3) RQ3. How does the clustering approach compare to the existing SP estimation approaches?: To answer this question, we compare the best variant of our clustering approach-based model to two previous work, including a sate-of-the-art deep-learning-based model for SP estimation. Specifically, we compare the estimation accuracy of our approach to TF-IDF-SE and Deep-SE (see description of these approaches in Section IV-C).

B. Data

We use a large dataset of issues called the Tawosi dataset [44]. This dataset comprises of 26 projects, from 13 different open-source repositories. All projects included in this dataset use agile development methodologies. The dataset contains the title, description, type, component, and the human estimated SP for 31,960 issues in total. Table I shows the descriptive statistics of the projects in the Tawosi dataset. Due to space limitations, full details about how the data was collected and its characteristic can be found elsewhere [44].

The issues are in ascending order of their creation time in the dataset. We split the issues in each project for training, validation, and testing subsets with a ratio of 60%:20%:20%, using the older issues for training and newer issues for testing.

C. Benchmarks

This section provides an overview of the story point estimation techniques that are used for comparison; namely Deep-SE [14], TF-IDF-SE [11], and the Median, Mean, and Random Guessing (RG) baselines. Deep-SE is the current state-of-the-art in story point estimation. Both Deep-SE and TF-IDF-SE leverage the similarity between the target issue and the previously estimated issues to produce an estimation for the target issue. TF-IDF-SE relies on the term frequency-inverse document frequency (TF-IDF) feature model, while, Deep-SE uses advanced techniques in deep-learning to exploit the semantic similarity between user stories. On the other hand, the three baselines are agnostic to any information from the issue description and only use the distribution characteristics (i.e., mean, median, or random sampling) of the previous estimations to estimate new SPs.

1) Deep-SE: is an end-to-end deep learning model to estimate story point score of a software issue, proposed by Choetkiertikul et al. [14]. This model is composed of four components: (1) Word Embedding, (2) Document representation using Long-Short Term Memory (LSTM) [34], [35], (3) Deep representation using Recurrent Highway Network (RHWN) [36], and (4) Differentiable Regression. The first component converts each word in the title and description of issues into a fixed-length vector (i.e., word embedding). These word vectors are then fed to the LSTM layer which in turn computes a vector representation for the story. Then, the RHWN accepts the document vector as input and transforms it multiple times, until a final vector which represents the text is produced as output. This final vector serves as input to the regressor, which predicts the output story point. The word embedding and LSTM layers are pre-trained without using SP values to come up with a proper parameter initialization for the main deep structure (referred to as the pre-trained language models). We used the implementation of Deep-SE provided by Choetkiertikul et al., in their replication package [41].

TABLE I
DESCRIPTIVE STATISTICS OF THE TAWOSI DATASET [44] USED IN THIS STUDY.

Repository	Project	Key	#Issues	Story Point				
				Min	Max	Mean	Median	Std
Apache	Mesos	MESOS	1,513	0	13	3.15	3	2.14
Appcelerator	Alloy	ALOY	241	0	13	3.71	3	2.32
	Appcelerator Studio	TISTUD	2,794	0	40	5.48	5	3.23
	Aptana Studio	APSTUD	476	0	100	7.93	8	7.19
	Command-Line Interface	CLI	293	0	13	3.18	3	2.30
	Daemon	DAEMON	205	1	13	5.58	5	3.76
	Documentation	TIDOC	1,005	0	40	3.58	2	3.68
	Titanium	TIMOB	3,915	0	20	4.68	5	3.32
Atlassian	Clover	CLOV	336	0	100	5.33	2	11.03
	Confluence Cloud	CONFCLOUD	234	0	13	2.91	2	2.24
	Confluence Server and Data Center	CONFSERVER	456	0	13	3.12	3	1.93
DNNSoftware	DNN	DNN	2,064	0	100	2.05	2	2.56
DuraSpace	Duracloud	DURACLOUD	310	0	20	1.72	1	1.70
Hyperledger	Fabric	FAB	303	0	40	2.69	2	3.20
	Sawtooth	STL	206	0	5	2.09	2	1.19
Lsstcorp	Data Management	DM	5,381	0	100	3.05	2	6.87
MongoDB	Compass	COMPASS	260	1	8	3.55	3	1.85
	Core Server	SERVER	519	0	20	2.58	2	2.40
	Evergreen	EVG	2,824	0	8	1.43	1	0.86
Moodle	Moodle	MDL	1,394	0	100	11.80	5	18.81
MuleSoft	Mule	MULE	2,935	0	13	3.88	3	3.46
Sonatype	Sonatype's Nexus	NEXUS	1,425	0	40	1.70	1	1.82
Spring	SpringXD	XD	811	0	20	3.16	3	2.56
Talendforge	Talend Data Preparation	TDP	471	0	13	2.31	2	1.84
	Talend Data Quality	TDQ	859	0	40	6.01	5	4.66
	Talend ESB	TESB	730	0	13	2.13	2	1.45
Total			31,960					

2) TF-IDF-SE: is proposed by Porru et al. [11], and treats SP estimation as a classification problem. It uses the title, description and length of issues, alongside their type and component, to build a machine learning classifier for story point estimation. They separate the code snippets (if any) from natural language text in the issue description and analyse two chunks separately to extract TF-IDF features. They concatenate three feature sets, two extracted from text and code chunks respectively, and the third is a one-hot representation of issues' type and components, before reducing the dimension using feature selection. The selected features are then fed to a Support Vector Machine (SVM) to classify issues into SP classes. Choetkiertikul et al. also provided the implementation for TF-IDF-SE in their replication package [41], which we employed in this study.

3) Random Guessing: is a naïve method that simply assigns story point of a randomly selected issue from past to the target issue [30]. Comparison over random guessing serves as a sanity check for a proposed effort estimation technique. Formally, random guessing predicts a story point value y for the target case $issue_t$ by uniformly randomly sampling over all the remaining $n-1$ cases and taking $y = r$; where r is the story point for the randomly drawn $issue_r$ from $1...n \mid issue_r \neq issue_t$. This method does not need any parameter estimation and any prediction system is expected to outperform it over time, otherwise, the prediction system is not using any target case information.

4) Mean and Median Estimators: are two baseline benchmarks commonly used for effort estimation techniques [32], [37], [38]. Specifically, the mean or median story point of the past issues is used as the predicted story point for a new issue.

D. Evaluation Measures

Similar to previous studies on software effort estimation, we use measurements that are built upon the error (or absolute error) between the predicted value and the actual value. These measures (defined in Equations 1, 2 and 3) have been found in previous work to be unbiased towards under- or over-estimations [6], [14], [30]–[32]. These measure are the Mean Absolute Error (MAE), the Median Absolute Error (MdAE), and the Standard Accuracy (SA).

Across n issues, the MAE and MdAE of prediction for a project are computed as follows:

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

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

where $actual_i$ is the actual SP, $predicted_i$ is the predicted SP for the i^{th} issue, and n is the number of issues in the project.

SA was recommended by Shepperd and MacDonell [30] as a standard measure to compare multiple prediction models against each other. It is based on MAE and 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 take a value in the range $[0, 1]$. An SA value closer to zero means that the predictor p_i is not performing much better than random guessing [30]. A negative SA value means that the prediction model is outperformed by random guessing. For a high-performance prediction model, MAE and $MdAE$ should be lower, whereas SA should be higher.

E. Statistical Analysis

To check if the difference in the results achieved by two methods is statistically significant, we performed a non-parametric statistical test. Specifically, the Wilcoxon Ranked-Sum test (a.k.a. Mann–Whitney U test) [40] with confidence limit at $\alpha = 0.05$, corrected with Bonferroni, is applied on the distribution of the absolute errors produced by the methods under investigation. We tested the hypothesis:

> H_0: *The absolute errors produced by the prediction model P_i are higher than those produced by the prediction model P_j.*

We used one-way Wilcoxon test; hence, if the test rejects the null hypothesis, the alternative hypothesis is accepted:

> H_1: *The absolute errors produced by the P_i are lower than those provided by the P_j.*

As done in previous work [45]–[48], we use the win-loss-tie counting to summarise the results of the Wilcoxon test, as follows: if the distribution i is statistically significantly better (less) than j according to the Wilcoxon test we update win_i and $loss_j$, otherwise we increment tie_i and tie_j.

To measure the effect size of the difference, we use Vargha Delaney's $\hat{A}_{12}$ measure [33], which is a standardised non-parametric effect size measurement, to assess the effect size of the difference between two methods [32], [33]. For two algorithms 1 and 2, $\hat{A}_{12}$ measures the probability of 1 performing better than 2 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. Based on Equation (4), if two algorithms are equally good, $\hat{A}_{12} = 0.5$. Respectively, $\hat{A}_{12}$ higher than 0.5 means that the first algorithm is more likely to produce better predictions. The effect size is considered small for $0.6 \leq \hat{A}_{12} < 0.7$, medium for $0.7 \leq \hat{A}_{12} \leq 0.8$, and large for $\hat{A}_{12} \geq 0.8$, although these thresholds are not definitive [6].

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

V. Results

This section presents the results of our empirical study for each of the proposed research questions.

A. RQ1. Does clustering of issue reports based on their textual similarities help accurately estimate story points?

Identifying the best LHC-SE model: As described in Sections III-B and III-C, our study explores the use of LHC-SE with three different cluster forming strategies and three estimators, for a total nine different LHC-SE estimation models. Before comparing LHC-SE to the baselines, we study which combination of cluster forming and estimation strategies works best with LHC-SE. To this end, we compare these nine strategies against each other based on the Wilcoxon Rank-Sum test and summarise the results using the win-loss-tie approach explained in Section IV-E. The results are shown in Table II, where for each method, the rows and columns represent the nine strategies, and each cell contains the number of cases (out of 26 projects) the strategy in the row won/lost/tied against the strategy in the column. Specifically, a win is counted if the strategy in the row produces statistically significantly lower absolute errors than the strategy in the column.

Based on the results shown in Table II, we can observe that the MAE-based k-selection strategy with Cluster Median estimation model wins most of the times (117 wins, 1 loss, and 90 ties). Thus, we select this combination of strategy and estimation model for LHC-SE to compare with the baselines.

Sanity Check: Table III shows the MAE and SA values achieved by LHC-SE and the baselines. The MdAE values are also reported, for completeness.

We can observe that 24 out of 26 SA values for LHC-SE are positive, which means that LHC-SE outperforms the Random Guessing (RG) baseline in 24 cases (exceptions are the STL and DURACLOUD projects). For all these 24 cases, the difference between the absolute errors produced by LHC-SE and RG is statistically significant in favour of LHC-SE, and 12 cases also showed a large or medium effect size, while the remaining a small or negligible one.

LHC-SE achieves a good performance against Mean baseline as well. It outperforms the Mean estimator in 20 cases, while under-performing in only 6 cases. From these 20 cases, the improvement is statistically significant in 18 cases, with large effect size in 5 cases and small or negligible one in the rest. However, against the Median estimator, LHC-SE performs rather poorly, albeit with a negligible effect size. Although LHC-SE outperforms the Median estimator in 10 cases, in the remaining 16 cases the Median estimator outperforms LHC-SE. Nonetheless, the results of the Wilcoxon test reveal that the difference in the estimation performance of these two methods is statistically significant in only three

TABLE II

RQ1 and RQ2: Win-Loss-Tie results comparing the nine different combinations of three cluster building methods and three estimation strategies for each of the three LHC-SE-based variants. The best strategy for each variant is highlighted.

RQ				Win/Loss/Tie									
	Method			LHC-SE									
		k-selection strategy		MAE-based			MdAE-based			Silhouette-based			
			Estimator	Closest Point	Cluster Mean	Cluster Median	Closest Point	Cluster Mean	Cluster Median	Closest Point	Cluster Mean	Cluster Median	Sum
RQ1	LHC-SE	MAE-based	Closest Point		3/8/15	0/20/6	0/0/26	3/9/14	0/20/6	0/0/26	2/6/18	0/14/12	8/77/123
			Cluster Mean	8/3/15		0/17/9	8/3/15	0/5/21	0/17/9	8/3/15	0/5/21	0/13/13	24/66/118
			Cluster Median	20/0/6	17/0/9		20/0/6	16/0/10	1/1/24	20/0/6	18/0/8	5/0/21	**117/1/90**
		MdAE-based	Closest Point	0/0/26	3/8/15	0/20/6		3/9/14	0/20/6	0/0/26	2/6/18	0/14/12	8/77/123
			Cluster Mean	9/3/14	5/0/21	0/16/10	9/3/14		0/16/10	9/3/14	0/5/21	0/13/13	32/59/117
			Cluster Median	20/0/6	17/0/9	1/1/24	20/0/6	16/0/10		20/0/6	16/0/10	3/0/23	113/1/94
		Silhouette-based	Closest Point	0/0/26	3/8/15	0/20/6	0/0/26	3/9/14	0/20/6		2/6/18	0/14/12	8/77/123
			Cluster Mean	6/2/18	5/0/21	0/18/8	6/2/18	5/0/21	0/16/10	6/2/18		0/12/14	28/52/128
			Cluster Median	13/0/13	13/0/13	0/5/21	13/0/13	13/0/13	0/3/23	13/0/13	12/0/14		77/8/123
	Method			LHC$_{TC}$-SE									
		k-selection strategy		MAE-based			MdAE-based			Silhouette-based			
			Estimator	Closest Point	Cluster Mean	Cluster Median	Closest Point	Cluster Mean	Cluster Median	Closest Point	Cluster Mean	Cluster Median	Sum
RQ2	LHC$_{TC}$-SE	MAE-based	Closest Point		3/6/17	0/16/10	1/0/25	4/6/16	0/17/9	1/0/25	3/5/18	0/17/9	12/67/129
			Cluster Mean	6/3/17		0/13/13	5/5/16	2/3/21	1/14/11	5/4/17	5/2/19	1/14/11	25/58/125
			Cluster Median	16/0/10	13/0/13		16/0/10	16/0/10	3/1/22	15/0/11	18/0/8	4/1/21	**101/2/105**
		MdAE-based	Closest Point	0/1/25	5/6/15	0/16/10		7/6/13	1/17/8	0/0/26	5/6/15	0/16/10	18/68/122
			Cluster Mean	6/4/16	3/2/21	0/16/10	5/7/14		1/15/10	5/6/15	7/3/16	1/15/10	28/68/112
			Cluster Median	17/0/9	14/1/11	1/3/22	17/1/8	15/1/10		17/1/8	17/1/8	2/1/23	100/9/99
		Silhouette-based	Closest Point	0/1/25	4/5/17	0/15/11	0/0/26	6/5/15	1/17/8		4/5/17	0/17/9	15/65/128
			Cluster Mean	5/3/18	2/5/19	0/18/8	5/5/16	3/7/16	1/17/8	5/4/17		1/17/8	22/76/110
			Cluster Median	17/0/9	14/1/11	1/4/21	16/0/10	15/1/10	1/2/23	17/0/9	17/1/8		98/9/101
	Method			LHC$_{TC+TFIDF}$-SE									
		k-selection strategy		MAE-based			MdAE-based			Silhouette-based			
			Estimator	Closest Point	Cluster Mean	Cluster Median	Closest Point	Cluster Mean	Cluster Median	Closest Point	Cluster Mean	Cluster Median	Sum
RQ2	LHC$_{TC+TFIDF}$-SE	MAE-based	Closest Point		4/13/9	0/20/6	0/0/26	4/12/10	1/21/4	0/0/26	5/12/9	1/20/5	15/98/95
			Cluster Mean	13/4/9		0/17/9	13/4/9	5/5/16	1/18/7	13/4/9	5/2/19	2/17/7	52/71/85
			Cluster Median	20/0/6	17/0/9		20/0/6	17/0/9	0/1/25	20/0/6	17/0/9	2/2/22	113/3/92
		MdAE-based	Closest Point	0/0/26	4/13/9	0/20/6		4/12/10	1/21/4	0/0/26	5/12/9	1/20/5	15/98/95
			Cluster Mean	12/4/10	5/5/16	0/17/9	12/4/10		1/18/7	12/4/10	7/4/15	2/17/7	51/73/84
			Cluster Median	21/1/4	17/1/8	1/0/25	21/1/4	18/1/7		21/1/4	18/1/7	2/1/23	**119/7/82**
		Silhouette-based	Closest Point	0/0/26	4/13/9	0/20/6	0/0/26	4/12/10	1/21/4		5/12/9	1/20/5	15/98/95
			Cluster Mean	12/5/9	2/5/19	0/17/9	12/5/9	4/7/15	1/18/7	12/5/9		1/19/6	44/81/83
			Cluster Median	20/1/5	16/2/8	2/2/22	20/1/5	17/2/7	1/2/23	20/1/5	19/1/6		115/12/81

cases, one in favor of LHC-SE and two in favor of the Median baseline.

These results show that LHC-SE outperforms RG and Mean baselines in the majority of the cases, but emerges shoulder-to-shoulder with Median baseline. This motivates checking whether augmenting the feature set of LHC-SE increases its accuracy (RQ2).

Answer to RQ1: *LHC-SE easily outperforms Random Guessing and Mean baselines, and performs similarly to the Median baseline.*

B. RQ2. Can additional features help improve estimation accuracy of the clustering approach?

To answer this question, we compare the base LHC-SE model from RQ1 to two other variants (i.e., LHC$_{TC}$-SE which incorporates issue type and components, and LHC$_{TC+TFIDF}$-SE which incorporates the aforementioned in addition to TF-IDF scores for each issue).

Identifying the best strategies: Similar to RQ1, we first identify which combination of cluster forming and estimation strategies works best with each of the two additional variants. The middle and last rows of Table II show the win-loss-tie scores of the two variants (i.e., LHC$_{TC}$-SE and LHC$_{TC+TFIDF}$-SE) for different k-selection and estimation strategies, with respect to their Wilcoxon test results. We can observe that the best combination for LHC$_{TC}$-SE is the MAE-based k-selection with Cluster Median estimation, achieving the highest score (101 wins, 2 losses, and 105 ties). Whereas, for LHC$_{TC+TFIDF}$-SE the MdAE-based k-selection with Cluster Median estimator scores highest.

Comparing LHC-SE variants: Using the strategy that works the best for each variant, we compare the three LHC-based variants (each with their best performing strategy) in Table IV. As we can see, the three models score very close to one another (they draw a tie in almost all the cases). Specifically, LHC-SE and LHC$_{TC}$-SE perform similarly (each beats the other on two projects, and are tie on the others). However, LHC$_{TC}$-SE is better than LHC$_{TC+TFIDF}$-SE in more cases, and therefore it scores the highest number of wins among the three. So, we select LHC$_{TC}$-SE for the comparison against existing SP estimation approaches (RQ3).

Answer to RQ2: *Using type, component(s) and report length of issues, in addition to their LDA topics, help LHC$_{TC}$-SE perform better.*

TABLE III
RQ1 and RQ3: MAE, MdAE and SA values achieved by LHC-SE, LHC$_{TC}$-SE, Deep-SE, TF-IDF-SE, and the Mean and Median baselines. The best values per method and per project are printed in bold face.

Project	Method	MAE	MdAE	SA	Project	Method	MAE	MdAE	SA	Project	Method	MAE	MdAE	SA
MESOS	LHC-SE	1.34	**1.00**	34.38	CONFCLOUD	LHC-SE	1.34	1.00	40.41	SERVER	LHC-SE	**0.85**	1.00	**59.47**
	LHC$_{TC}$-SE	**1.33**	**1.00**	**34.63**		LHC$_{TC}$-SE	1.37	1.00	39.04		LHC$_{TC}$-SE	**0.85**	1.00	**59.47**
	Deep-SE	1.34	1.12	34.07		Deep-SE	1.48	**0.93**	33.89		Deep-SE	0.89	**0.71**	57.60
	TF-IDF-SE	1.34	**1.00**	34.38		TF-IDF-SE	**1.33**	1.00	40.86		TF-IDF-SE	0.93	1.00	55.88
	Mean	1.37	1.08	32.72		Mean	1.49	1.23	33.65		Mean	1.56	1.86	25.99
	Median	1.34	**1.00**	34.38		Median	**1.33**	1.00	**40.87**		Median	**0.85**	1.00	59.46
ALOY	LHC-SE	1.84	2.00	26.57	CONFSERVER	LHC-SE	0.96	1.00	49.64	MDL	LHC-SE	6.31	7.00	57.30
	LHC$_{TC}$-SE	2.28	2.00	9.01		LHC$_{TC}$-SE	0.96	1.00	49.64		LHC$_{TC}$-SE	6.31	7.00	57.30
	Deep-SE	1.51	**1.28**	39.67		Deep-SE	**0.91**	**0.64**	**52.28**		Deep-SE	**3.55**	**2.77**	**76.00**
	TF-IDF-SE	**1.44**	2.00	**42.53**		TF-IDF-SE	0.96	1.00	49.64		TF-IDF-SE	6.31	7.00	57.30
	Mean	2.23	2.17	10.84		Mean	1.35	1.45	29.17		Mean	14.54	15.23	1.58
	Median	**1.44**	2.00	**42.53**		Median	0.96	1.00	49.64		Median	6.31	7.00	57.30
APSTUD	LHC-SE	4.14	3.00	30.20	DNN	LHC-SE	**0.71**	1.00	**42.60**	MULE	LHC-SE	2.27	2.00	37.11
	LHC$_{TC}$-SE	**3.99**	3.00	**32.81**		LHC$_{TC}$-SE	**0.71**	1.00	**42.60**		LHC$_{TC}$-SE	2.60	3.00	28.16
	Deep-SE	4.31	2.70	27.37		Deep-SE	0.72	**0.69**	41.69		Deep-SE	**2.24**	**1.68**	37.95
	TF-IDF-SE	**3.99**	3.00	**32.81**		TF-IDF-SE	0.79	1.00	36.13		TF-IDF-SE	3.58	2.00	0.81
	Mean	4.00	**2.49**	32.72		Mean	0.80	0.88	35.28		Mean	2.79	3.18	22.68
	Median	**3.99**	3.00	**32.81**		Median	**0.71**	1.00	**42.60**		Median	**2.24**	2.00	**38.05**
CLI	LHC-SE	1.87	2.00	29.32	FAB	LHC-SE	0.67	1.00	69.75	NEXUS	LHC-SE	1.14	1.00	22.52
	LHC$_{TC}$-SE	**1.76**	2.00	33.35		LHC$_{TC}$-SE	**0.65**	1.00	**70.47**		LHC$_{TC}$-SE	1.22	1.00	16.88
	Deep-SE	**1.76**	**1.30**	**33.44**		Deep-SE	0.86	**0.71**	61.06		Deep-SE	**1.08**	0.88	**26.56**
	TF-IDF-SE	2.98	3.00	-12.84		TF-IDF-SE	1.10	1.00	50.31		TF-IDF-SE	1.17	1.00	20.68
	Mean	2.14	2.61	18.93		Mean	1.19	1.10	46.21		Mean	1.11	**0.58**	24.69
	Median	1.77	2.00	33.04		Median	0.67	1.00	69.75		Median	1.17	1.00	20.68
DAEMON	LHC-SE	2.81	3.00	32.09	STL	LHC-SE	1.28	1.00	-6.77	XD	LHC-SE	1.54	**1.00**	39.53
	LHC$_{TC}$-SE	**2.74**	3.00	**33.81**		LHC$_{TC}$-SE	0.95	1.00	20.41		LHC$_{TC}$-SE	1.50	**1.00**	40.85
	Deep-SE	3.29	**2.00**	20.55		Deep-SE	1.18	1.12	1.91		Deep-SE	**1.45**	1.16	**43.06**
	TF-IDF-SE	**2.74**	3.00	**33.81**		TF-IDF-SE	**0.84**	**0.00**	**30.12**		TF-IDF-SE	2.01	2.00	20.82
	Mean	2.75	2.75	33.53		Mean	0.97	1.02	19.32		Mean	1.65	1.72	34.89
	Median	**2.74**	3.00	**33.81**		Median	0.95	1.00	20.41		Median	1.55	**1.00**	39.05
TIDOC	LHC-SE	2.79	**1.00**	23.48	DM	LHC-SE	1.56	1.00	53.87	TDP	LHC-SE	1.00	1.00	37.08
	LHC$_{TC}$-SE	3.65	2.00	-0.30		LHC$_{TC}$-SE	1.52	1.00	54.94		LHC$_{TC}$-SE	1.03	1.00	35.44
	Deep-SE	**2.72**	1.19	**25.35**		Deep-SE	1.61	**0.89**	52.41		Deep-SE	**0.99**	**0.81**	37.69
	TF-IDF-SE	3.03	**1.00**	16.69		TF-IDF-SE	**1.49**	1.00	**55.71**		TF-IDF-SE	**0.99**	1.00	**37.74**
	Mean	2.99	2.59	18.00		Mean	2.60	2.43	22.83		Mean	1.17	1.38	26.26
	Median	2.77	**1.00**	24.03		Median	1.61	1.00	52.19		Median	**0.99**	1.00	**37.74**
TIMOB	LHC-SE	2.53	2.00	30.70	DURACLOUD	LHC-SE	1.25	1.00	-9.65	TDQ	LHC-SE	3.52	3.00	27.60
	LHC$_{TC}$-SE	2.48	2.00	32.12		LHC$_{TC}$-SE	0.68	1.00	39.94		LHC$_{TC}$-SE	2.92	3.00	40.01
	Deep-SE	**2.41**	**1.81**	**33.90**		Deep-SE	0.68	**0.58**	39.90		Deep-SE	**2.47**	**2.23**	**49.14**
	TF-IDF-SE	2.53	2.00	30.70		TF-IDF-SE	0.68	1.00	39.94		TF-IDF-SE	5.05	5.00	-3.95
	Mean	2.55	**1.81**	30.23		Mean	**0.67**	0.85	**41.13**		Mean	4.20	3.82	13.65
	Median	2.53	2.00	30.70		Median	0.68	1.00	39.94		Median	2.88	3.00	40.72
TISTUD	LHC-SE	**1.51**	2.00	**51.89**	COMPASS	LHC-SE	1.38	2.00	28.54	TESB	LHC-SE	0.99	1.00	32.56
	LHC$_{TC}$-SE	**1.51**	2.00	**51.89**		LHC$_{TC}$-SE	**1.30**	**1.00**	**32.46**		LHC$_{TC}$-SE	1.04	1.00	29.31
	Deep-SE	1.63	**1.38**	48.08		Deep-SE	1.63	1.34	15.25		Deep-SE	1.15	**0.73**	21.36
	TF-IDF-SE	**1.51**	2.00	**51.89**		TF-IDF-SE	1.38	2.00	28.54		TF-IDF-SE	**0.97**	1.00	**33.95**
	Mean	2.01	2.16	35.93		Mean	1.48	1.63	23.05		Mean	0.99	0.99	32.71
	Median	**1.51**	2.00	**51.89**		Median	1.38	2.00	28.54		Median	0.98	1.00	33.02
CLOV	LHC-SE	3.88	2.00	46.35	EVG	LHC-SE	**0.60**	1.00	**22.69**					
	LHC$_{TC}$-SE	4.23	1.50	41.55		LHC$_{TC}$-SE	0.62	1.00	19.97					
	Deep-SE	**3.78**	1.05	**47.73**		Deep-SE	0.63	**0.54**	19.39					
	TF-IDF-SE	4.04	**1.00**	44.15		TF-IDF-SE	0.69	1.00	10.67					
	Mean	5.93	5.30	18.06		Mean	0.68	0.56	12.98					
	Median	4.01	2.00	44.55		Median	0.69	1.00	10.67					

TABLE IV
RQ2: Win-Loss-Tie summary of the Wilcoxon test results comparing the LHC-SE variants with their respective best strategies from Table II. The best variant is highlighted.

	Win/Loss/Tie			
Method	LHC-SE	LHC$_{TC}$-SE	LHC$_{TC+TFIDF}$-SE	Sum
LHC-SE		2/2/22	1/2/23	3/4/45
LHC$_{TC}$-SE	2/2/22		2/1/23	**4/3/45**
LHC$_{TC+TFIDF}$-SE	2/1/23	1/2/23		3/3/46

C. RQ3. How does the clustering approach compare to the existing SP estimation approaches?

As presented in RQ2, LHC$_{TC}$-SE was found to be the best performing variant of the three investigated models. Hence, we compare this variant against state-of-the-art.

Table III shows the MAE and SA values achieved by LHC$_{TC}$-SE, Deep-SE, TF-IDF-SE, and the Mean, Median, and Random Guessing (RG) baselines (the MdAE values are also reported, for completeness). We can observe that LHC$_{TC}$-SE achieves a better (lower) MAE than Deep-SE in 14 out of 26 cases, while Deep-SE achieves a better MAE in only 12 cases. LHC$_{TC}$-SE outperforms TF-IDF-SE in 15 cases, and under performs in the remaining 11 cases. LHC$_{TC}$-SE also achieves better MAE values than those achieved by Mean and RG in 21 and 25 cases, respectively. It achieves better MAEs than the Median estimator, in 14 cases, and slightly worse in 12.

It is worth noting that in some cases the MAE values are very close (e.g. the MULE project in Table III), showing that achieving a lower MAE does not guarantee that a method performs statistically significantly better than the other. For this reason, we also provide the p-values and effect sizes of the statistical tests performed on LHC$_{TC}$-SE against the other methods per project in Table V and, then summarise these results as win-loss-tie in Table VI.

Based on the results reported in Table V, we observe that LHC$_{TC}$-SE performs statistically significantly better than Deep-SE for five projects (i.e., TISTUD, FAB, DM, COMPASS, and EVG), however, the effect size for all five cases is negligible. Similarly, LHC$_{TC}$-SE performs statistically significantly better than the Median estimator in two projects (i.e., DM and EVG), but the effect size for both cases is negligible. Compared to TF-IDF-SE, LHC$_{TC}$-SE showed statistically significant improvement for five projects (i.e., CLI, FAB, EVG, MULE, and TDQ), in two cases with a medium effect size, in one case with a small one, and in the remaining two cases with a negligible one. Against the Mean estimator, LHC$_{TC}$-SE shows a significant improvement. Particularly, from 19 projects for which LHC$_{TC}$-SE produces statistically significantly better estimations, in four cases the difference shows a strong effect size, in one case the effect size is medium, and for the remaining 14 cases seven show medium and seven show negligible effect sizes. Finally, compared to the RG baseline, LHC$_{TC}$-SE shows statistically significant difference for all projects but one. Among the 25 projects which LHC$_{TC}$-SE outperforms RG, half (12 cases) show a strong effect size, seven cases show a medium, three cases a small, and one case a negligible effect size.

Based on the win-tie-loss summary (Table VI), we can conclude that LHC$_{TC}$-SE scores are very close to Deep-SE and Median estimator, though it is ahead by two and one wins, respectively. Considering the scores achieved by LHC$_{TC}$-SE against the other methods (see first row Table VI), we can also observe that LHC$_{TC}$-SE never wins less than it loses to the other methods.

Overall, these results show that our proposed method outperforms RG and Mean baselines statistically significantly, and matches the accuracy of the state-of-the-art, while slightly enhancing it in some cases, it does not perform worse in most of the cases. It also performs as good as the Median estimator.

Answer to RQ3: *LHC$_{TC}$-SE matches the accuracy of the state-of-the-art, while slightly enhancing it in some cases, it does not perform worse in most of the cases.*

TABLE V
RQ3: WILCOXON TEST RESULTS (WITH VARGHA-DELANEY EFFECT SIZE IN BRACKETS) COMPARING LHC$_{TC}$-SE AGAINST EACH OF THE PREVIOUS WORK AND THE BASELINE METHODS.

Project	LHC$_{TC}$-SE vs.				
	Deep-SE	TF-IDF-SE	Mean	Median	Random
MESOS	0.147 (0.52)	0.420 (0.50)	0.001 (0.57)	0.420 (0.50)	<0.001 (0.73)
ALOY	0.992 (0.36)	0.999 (0.33)	0.435 (0.51)	0.999 (0.33)	0.167 (0.56)
APSTUD	0.186 (0.54)	0.501 (0.50)	0.370 (0.51)	0.501 (0.50)	<0.001 (0.73)
CLI	0.736 (0.47)	<0.001 (0.74)	0.030 (0.60)	0.466 (0.50)	<0.001 (0.75)
DAEMON	0.344 (0.53)	0.502 (0.50)	0.629 (0.48)	0.502 (0.50)	<0.001 (0.77)
TIDOC	0.826 (0.47)	0.869 (0.47)	0.002 (0.58)	0.873 (0.47)	<0.001 (0.70)
TIMOB	0.563 (0.50)	0.121 (0.52)	0.048 (0.52)	0.121 (0.52)	<0.001 (0.73)
TISTUD	<0.001 (0.60)	0.500 (0.50)	<0.001 (0.63)	0.500 (0.50)	<0.001 (0.84)
CLOV	0.920 (0.43)	0.949 (0.42)	<0.001 (0.75)	0.301 (0.53)	<0.001 (0.81)
CONFCLOUD	0.497 (0.50)	0.552 (0.49)	0.036 (0.60)	0.552 (0.49)	<0.001 (0.81)
CONFSERVER	0.132 (0.55)	0.501 (0.50)	<0.001 (0.65)	0.501 (0.50)	<0.001 (0.77)
DNN	0.630 (0.49)	0.386 (0.51)	<0.001 (0.69)	0.500 (0.50)	<0.001 (0.82)
FAB	0.003 (0.64)	0.002 (0.64)	<0.001 (0.83)	0.435 (0.51)	<0.001 (0.95)
STL	0.110 (0.58)	0.951 (0.41)	0.007 (0.65)	0.502 (0.50)	0.003 (0.67)
DM	<0.001 (0.56)	1.000 (0.46)	<0.001 (0.83)	<0.001 (0.54)	<0.001 (0.93)
DURACLOUD	0.052 (0.58)	0.501 (0.50)	0.181 (0.55)	0.501 (0.50)	<0.001 (0.69)
COMPASS	0.043 (0.60)	0.403 (0.51)	0.112 (0.57)	0.403 (0.51)	0.004 (0.65)
EVG	0.034 (0.53)	0.008 (0.54)	0.045 (0.53)	0.008 (0.54)	0.015 (0.54)
SERVER	0.413 (0.51)	0.481 (0.50)	<0.001 (0.80)	0.422 (0.51)	<0.001 (0.91)
MDL	1.000 (0.17)	0.500 (0.50)	<0.001 (1.00)	0.500 (0.50)	<0.001 (1.00)
MULE	1.000 (0.41)	<0.001 (0.57)	<0.001 (0.59)	1.000 (0.41)	<0.001 (0.77)
NEXUS	0.721 (0.49)	0.892 (0.47)	0.897 (0.47)	0.892 (0.47)	<0.001 (0.59)
XD	0.626 (0.49)	0.056 (0.55)	0.005 (0.58)	0.774 (0.48)	<0.001 (0.84)
TDP	0.613 (0.49)	0.564 (0.49)	0.016 (0.59)	0.564 (0.49)	<0.001 (0.82)
TDQ	0.995 (0.42)	<0.001 (0.79)	<0.001 (0.67)	0.516 (0.50)	<0.001 (0.84)
TESB	0.105 (0.54)	0.817 (0.47)	0.056 (0.55)	0.753 (0.48)	<0.001 (0.67)

TABLE VI
RQ3: WIN-LOSS-TIE SUMMARY OF THE WILCOXON TEST RESULTS COMPARING LHC$_{TC}$-SE AGAINST EACH OF THE PREVIOUS WORK AND THE BASELINE METHODS. THE BEST METHOD IS HIGHLIGHTED

	Win/Loss/Tie						
Method	LHC$_{TC}$-SE	Deep-SE	TFI/DF-SE	Mean	Median	Random	Summary
LHC$_{TC}$-SE		5/4/17	5/3/18	19/0/7	2/2/22	25/0/1	**56/9/65**
Deep-SE	4/5/17		5/4/17	16/1/9	4/3/19	25/0/1	54/13/63
TF-IDF-SE	2/5/19	4/5/17		16/3/7	2/5/19	23/3/0	47/21/62
Mean	0/19/7	1/16/9	3/16/7		1/20/5	24/0/2	29/71/30
Median	2/2/22	3/4/19	5/3/18	20/1/5		25/1/0	55/11/64
Random	0/25/1	0/25/1	3/23/0	0/24/2	1/25/0		4/122/4

VI. DISCUSSION

Our results show that LDA is able to capture information latent in the issue-context to enable the clustering algorithm to form useful clusters for story point estimation.

In RQ1, we analysed LHC-SE, which solely uses LDA-generated posterior topic probabilities. This approach outperforms random guessing in all cases and the Mean baseline in 77% of the cases, based on the MAE values. However, the Median baseline performs as good as LHC-SE. We should note that the Median baseline is also performing better than all other methods investigated in this study (see Table VI), including the two previous work (TF-IDF-SE and Deep-SE)[1], though both are more sophisticated methods.

RQ2 results show that augmenting LDA-generated posterior topic probabilities with extra features from issue reports help the clustering algorithm to form a better clustering solution, thus improving the estimation accuracy, though marginally (see Table IV). In fact, the addition of TF-IDF weights to the feature set (i.e., LHC$_{TC+TFIDF}$-SE) showed improvements over LHC-SE, however the accuracy of LHC$_{TC+TFIDF}$-SE was slightly lower than those achieved by LHC$_{TC}$-SE. These results suggest that adding more discriminating attributes to the feature set can help the clustering algorithm form even higher quality clusters.

[1]We also note that the Median baseline outperforms the state-of-the-art approach (i.e., Deep-SE), which contradicts the results of the original study [14]. This motivated us to carry out a close replication of their original study. The results, which can be found elsewhere [44], confirm that Deep-SE is often outperformed by both the Median and Mean baselines.

Finally, RQ3 results reveal that LHC$_{TC}$-SE never performs worse than the other benchmarks (e.g., the Median baseline and Deep-SE). However, considering the level of complexity of the model, time and resources consumed to build it, and the interpretability of the models built, the Median estimator can be viewed as the more favourable model so far that can be used in practice.

We note that the fact that a naive estimation approach, such as the Median one, provides comparable and in some cases even better results than much more sophisticated techniques like deep-learning and LDA, strongly indicates that the research advances made so far are unsatisfactory. This also suggests that future research on story point estimation might need to pay more attention to the data rather than the estimation technique when building prediction models [21]. In fact, issue reports, especially in open-source projects, are not usually written in a structured or formal way; thus, they can be very noisy, and it is possible that by using tailored text pre-processing and data cleaning the accuracy of the proposed model can be improved. On the other hand, improving the quality of the user stories written by the authors of the issue report (for example by providing them with accurate guidelines or training) could yield less noisy data for model building; thus, improving the estimation accuracy. Besides, additional features can be extracted in order to aid prediction models in seeking more accurate estimations.

We believe that sharing these negative findings provides the research community with the knowledge needed to develop alternative strategies and evolve better solutions for story points estimation.

VII. Threats to Validity

Like previous studies, we use human-estimated story points as the ground truth, which might be biased. On one hand, this is mitigated by the clustering of similar issues, based on their description, hence augmenting the estimations of several human estimators. On the other hand, these values can be viewed as a placeholder that is used to test the model's ability to estimate SP based on historical issues' scores (wherever their origin might be). The model therefore can be trained on an unbiased target value, when it is available (for example, the real time spent on issue development). Currently, given the available dataset, our model can imitate human experts assisting them in their estimation, at its best.

To minimize threats to conclusion validity, we carefully selected unbiased accuracy performance measures and applied statistical tests to rule out small differences.

The dataset we used represents a wide range of real-world projects. However, we cannot claim that our dataset is representative of all software projects. All our projects are collected from open-source repositories, which can be different from industrial projects in many aspects. A key difference, that may affect the estimation of story points, is the behaviour of contributors, developers, and project stakeholders. It is also expected that in a commercial project setting issue reports may be written in a more disciplined environment, thus, providing more useful information and containing less noise. Therefore, further investigation for commercial projects from industrial software companies is needed to validate the conclusions made in this study.

VIII. Conclusion and Future Work

In this paper, we investigate a novel clustering-based model to estimate Story Point (SP), dubbed LHC-SE.

The idea behind LHC-SE is to leverage the similarity of issues, measured using the similarity of LDA-generated topic space of issue descriptions and agglomerative hierarchical clustering, to estimate the SP of a new issue based on the past most similar issues. This model works on the premise that clustering similar data points together helps reduce variance, and thus, increases the accuracy of any model built upon them.

To assess effectiveness of our proposal we have carried out a thorough empirical study benchmarking LHC-SE's performance against those of both baselines and state-of-the-art approaches for SP estimation on the largest corpus of open-source projects used in the literature to date.

The results showed that the use of LHC-SE allows us to achieve comparable results with the state-of-the-art (i.e., based on the Wilcoxon test results, it is statistically significantly better in 5 cases, worse in 4 and tie in the remaining 17 cases). On the other end, our results also surprisingly reveal that both LHC-SE and the state-of-the-art are comparable to some naive estimators such as simply assigning the median SP of previous issues, therefore their additional complexity does not seem warranted.

We hope that these negative findings encourage researchers to develop alternative strategies and evolve better ideas for story points estimation. In future work, we suggest investigating:

- More advanced data analysis and cleaning prior to model building.
- Utilizing other contextual text representation models recently introduced in NLP research, instead of LDA.
- Collecting and using additional effort-informative features available in, or derivable from, issue reports.
- Training machine learning methods on each cluster instead of baseline estimators used in this study.
- Exploring other distance measures for clustering instead of cosine similarity; and/or other clustering techniques instead of agglomerative hierarchical clustering.

Open Science

The data and code used in this study can be found at https://github.com/SOLAR-group/LHC-SE.git.

Acknowledgments

Vali Tawosi and Federica Sarro are supported by the ERC Advanced fellowship grant no. 741278 (EPIC).

REFERENCES

[1] M. Cohn, Agile estimating and planning. Pearson Education, 2005.

[2] A. Trendowicz and R. Jeffery, “Software project effort estimation,” Foundations and Best Practice Guidelines for Success, Constructive Cost Model–COCOMO pags, pp. 277–293, 2014.

[3] M. Usman, E. Mendes, F. Weidt, and R. Britto, “Effort estimation in agile software development: a systematic literature review,” in Proceedings of the 10th international conference on predictive models in software engineering, 2014, pp. 82–91.

[4] M. Jørgensen, “A review of studies on expert estimation of software development effort,” Journal of Systems and Software, vol. 70, no. 1-2, pp. 37–60, 2004.

[5] 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.

[6] V. Tawosi, F. Sarro, A. Petrozziello, and M. Harman. ”Multi-Objective Software Effort Estimation: A Replication Study.” IEEE Transactions on Software Engineering, 2021.

[7] M. Usman, J. Börstler, and K. Petersen. ”An effort estimation taxonomy for agile software development.” International Journal of Software Engineering and Knowledge Engineering 27, no. 04, 2017, pp. 641-674.

[8] A. Al-Subaihin, F. Sarro, S. Black, L. Capra, M. Harman, Y. Jia, and Y. Zhang. ”Clustering mobile apps based on mined textual features.” In Proceedings of the 10th ACM/IEEE international symposium on empirical software engineering and measurement, pp. 1-10. 2016.

[9] A. Al-Subaihin, F. Sarro, S. Black, and L. Capra. ”Empirical comparison of text-based mobile apps similarity measurement techniques.” Empirical Software Engineering 24, no. 6 (2019): 3290-3315.

[10] P. Abrahamsson, I. Fronza, R. Moser, J. Vlasenko, and W. Pedrycz, “Predicting development effort from user stories,” in 2011 International Symposium on Empirical Software Engineering and Measurement. IEEE, 2011, pp. 400–403.

[11] 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, 2016, pp. 1–10.

[12] E. Scott and D. Pfahl, “Using developers’ features to estimate story points,” in Proceedings of the 2018 International Conference on Software and System Process, 2018, pp. 106–110.

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

[14] 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.

[15] F. Ebrahimi, T. Miroslav, and M. Anas, ”Classifying Mobile Applications Using Word Embeddings.” in ACM Transactions on Software Engineering and Methodology, vol. 37, no. 111, pp. 30, 2021.

[16] F. Murtagh, and P. Legendre. ”Ward’s hierarchical agglomerative clustering method: which algorithms implement Ward’s criterion?.” Journal of classification 31, no. 3 (2014): 274-295.

[17] A. Starczewski, and A. Krzyżak. ”Performance evaluation of the silhouette index.” In International conference on artificial intelligence and soft computing, pp. 49-58. Springer, Cham, 2015.

[18] L. Minku, and S. Hou. ”Clustering dycom: An online cross-company software effort estimation study.” In Proceedings of the 13th International Conference on Predictive Models and Data Analytics in Software Engineering, pp. 12-21. 2017.

[19] V. K. Bardsiri, D. N. A. Jawawi, S. Z. M. Hashim, and E. Khatibi. ”Increasing the accuracy of software development effort estimation using projects clustering.” IET software 6, no. 6 (2012): 461-473.

[20] S. J. Huang, N. H. Chiu, and Y. J. Liu. ”A comparative evaluation on the accuracies of software effort estimates from clustered data.” Information and Software Technology 50, no. 9-10 (2008): 879-888.

[21] T. Menzies, Y. Yang, G. Mathew, B. Boehm, and J. Hihn. ”Negative results for software effort estimation.” Empirical Software Engineering 22, no. 5 (2017): 2658-2683.

[22] G. Scanniello, C. Gravino, A. Marcus, and T. Menzies. ”Class level fault prediction using software clustering.” In 2013 28th IEEE/ACM International Conference on Automated Software Engineering (ASE), pp. 640-645. IEEE, 2013.

[23] N. Bettenburg, M. Nagappan, and A.E. Hassan. ”Think locally, act globally: Improving defect and effort prediction models.” In 2012 9th IEEE Working Conference on Mining Software Repositories (MSR), pp. 60-69. IEEE, 2012.

[24] T. Menzies, A. Butcher, D. Cok, A. Marcus, L. Layman, F. Shull, B. Turhan, and T. Zimmermann. ”Local vs. Global Lessons for Defect Prediction and Effort Estimation,” IEEE Trans. Software Eng., preprint, published online Dec. 2012.

[25] X. Tan, X. Peng, S. Pan, and W. Zhao. ”Assessing software quality by program clustering and defect prediction.” In 2011 18th working conference on Reverse Engineering, pp. 244-248. IEEE, 2011.

[26] J. J. C. Gallego, D. Rodríguez, M. Á. Sicilia, M. G. Rubio, and A. G. Crespo. ”Software project effort estimation based on multiple parametric models generated through data clustering.” Journal of Computer Science and Technology 22, no. 3 (2007) 371-378.

[27] G. Nagpal, M. Uddin, and A. Kaur. ”Analyzing software effort estimation using k means clustered regression approach.” ACM SIGSOFT Software Engineering Notes 38, no. 1 (2013): 1-9.

[28] D. M. Blei, A. Y. Ng, and M. I. Jordan. ”Latent dirichlet allocation.” the Journal of machine Learning research 3 (2003): 993-1022.

[29] T. L. Griffiths, M. Steyvers. ”Finding scientific topics”. Proceedings of the National academy of Sciences 101, no. suppl 1 (2004): 5228-5235.

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

[31] 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.

[32] F. Sarro, A. Petrozziello, and M. Harman, “Multi-objective software effort estimation,” in 2016 IEEE/ACM 38th International Conference on Software Engineering (ICSE).

[33] 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.

[34] S. Hochreiter and J. Schmidhuber, “Long short-term memory,” Neural computation, vol. 9, no. 8, pp. 1735–1780, 1997.

[35] F. A. Gers, J. Schmidhuber, and F. Cummins, “Learning to forget: Continual prediction with lstm,” Neural computation 12, no. 10 (2000): 2451-2471.

[36] 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.

[37] 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.

[38] 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.

[39] M. Abadeer, and M. Sabetzadeh. ”Machine Learning-based Estimation of Story Points in Agile Development: Industrial Experience and Lessons Learned.” In 2021 IEEE 29th International Requirements Engineering Conference Workshops (REW), pp. 106-115. IEEE, 2021.

[40] J. Cohen. ”Statistical power analysis for the behavioral sciences.” Lawrence Earlbaum Associates, 2nd edition, 1988.

[41] “Source Code for Deep-SE and TF-IDF-SE GitHub.” [Online]. Available: https://github.com/SEAnalytics/datasets/tree/master/storypoint/IEEETSE2018.

[42] F. Filomena, E. Mendes, and F. Sarro. ”Web effort estimation: the value of cross-company data set compared to single-company data set.” PROMISE 2012: 29-38.

[43] E. Mendes, M. Kalinowski, D. Martins, F. Ferrucci, and F. Sarro. ”Cross- vs. within-company cost estimation studies revisited: an extended systematic review.” EASE 2014: 12:1-12:10.

[44] V. Tawosi, R. Moussa, and F. Sarro. ”Deep Learning for Agile Effort Estimation, Have We Solved the Problem Yet?” https://arxiv.org/abs/2201.05401, 2022.

[45] F. Sarro, and A. Petrozziello. ”Linear programming as a baseline for software effort estimation.” ACM transactions on software engineering and methodology (TOSEM) 27, no. 3 (2018): 1-28.

[46] E. Kocaguneli, T. Menzies, and J. W. Keung. ”On the value of ensemble effort estimation.” IEEE Transactions on Software Engineering 38, no. 6 (2011): 1403-1416.

[47] F. Sarro, F. Ferrucci, M. Harman, A. Manna, and J. Ren. ”Adaptive multi-objective evolutionary algorithms for overtime planning in software projects.” IEEE Transactions on Software Engineering 43, no. 10 (2017): 898-917.

[48] F. Sarro, M. Harman, Y. Jia, and Y. Zhang. ”Customer rating reactions can be predicted purely using app features.” In 2018 IEEE 26th International Requirements Engineering Conference (RE), pp. 76-87. IEEE, 2018.

[49] A. Lee, C. H. Cheng, and J. Balakrishnan. ”Software development cost estimation: integrating neural network with cluster analysis.” Information & Management 34, no. 1 (1998): 1-9.