2021 IEEE 29th International Requirements Engineering Conference Workshops (REW) | 978-1-6654-1898-0/21/$31.00 ©2021 IEEE | DOI: 10.1109/REW53955.2021.00022

Machine Learning-based Estimation of Story Points in Agile Development: Industrial Experience and Lessons Learned

Macarious Abadeer
Privacy Analytics Inc.,
an IQVIA company
Ottawa, Canada
macarious.abadeer@carleton.ca

Mehrdad Sabetzadeh
EECS, University of Ottawa
Ottawa, Canada
m.sabetzadeh@uottawa.ca

***Abstract*—Estimating story points is an important activity in agile software engineering. Story-point estimation enables software development teams to, among other things, better scope products, prioritize requirements, allocate resources and measure progress. Several machine learning techniques have been proposed for automated story-point estimation. However, most of these techniques use open-source projects for evaluation. There are important differences between open-source and commercial projects with respect to story authoring. The goal of this paper is to evaluate a state-of-the-art machine learning technique, known as Deep-SE [3], for estimating story points in a commercial project. Our dataset is comprised of 4,727 stories for a data anonymization product developed by a 27-member agile team at a healthcare data science company, IQVIA. Over this dataset, Deep-SE achieved a mean absolute error of 1.46, significantly better than three different baselines. Model performance nonetheless varied across stories, with the estimation error being larger for stories that had higher points. Our results further indicate that model performance is correlated with certain story characteristics such as the level of detail and the frequency of vague terms in the stories. An important take-away from our study is that, before organizations attempt to introduce machine learning-based estimation into agile development, they need to better embrace agile best practices, particularly in relation to story authoring and expert-based estimation.**

***Index Terms*—Agile Development, Story-point Estimation, Machine Learning**

I. Introduction

In agile development, story-point estimation is a process through which a value (story point) is assigned to a user story (story for short, hereafter) in order to indicate the level of effort anticipated for implementing that story [6]. Story-point estimation enables software development teams to better scope products, prioritize requirements, allocate resources, measure progress, forecast completion dates, and calculate the volume of stories that a team can handle, known as velocity [14]. Due to the rapidly changing scope of agile projects, poorly estimated stories can lead to uncertainty and increased costs [15].

Story points are typically estimated through planning sessions where a team reaches a consensus about how much effort a story requires in terms of story points. A team takes into account factors such as the complexity of the work required, the skill set of the team and the uncertainties that may arise [5]. Estimations are, however, often made by different team members who may assign low effort to certain stories that match their background. These differences in estimation may lead to inconsistencies and "guesstimates".

Choetkiertikul et al. [3] propose a machine learning technique, called *Deep-SE*, to automatically estimate story points using deep neural networks. The authors evaluate Deep-SE on 16 projects, all of which are open-source. In this paper, we set out to evaluate – in an empirically rigorous manner – the feasibility of applying Deep-SE for estimating story points in a (closed-source) commercial dataset with 4,727 stories.

There are important differences between open-source and commercial projects with respect to the nature of contributors and stakeholders [3]. For example, commercial projects tend to have internal jargon and acronyms that may not exist or may be less prevalent in the public space. Moreover, an open-source project usually has a large number of contributors with a high turnover rate [13], whereas a typical agile team in industry consists of less than 15 full-time members [10]. This can lead to more homogeneous story authoring in commercial projects. The iteration length can also be shorter in commercial projects due to marketing and delivery-timeline pressures [18]. Lastly, in the open-source world, issues can often remain open for longer [2]; this can increase the level of discussion and help add more details and context to open-source stories.

In addition to evaluating the performance of Deep-SE on a large, commercial dataset, we take preliminary steps towards identifying factors that can impact the quality of authored stories and expert estimates (inputs to machine learning) and consequently the accuracy of story-point predictions (output from machine learning) in industry. We reflect on the lessons learned from our investigation and provide recommendations for practitioners interested in automated story-point estimation.

The rest of this paper is organized as follows: Section II describes our industrial context. Section III discusses the related work. Section IV presents our adaptation of Deep-SE. Section V provides the design and results of our industrial evaluation. Section VI outlines the lessons learned. Section VII concludes the paper.

978-1-6654-1898-0/21/$31.00 ©2021 IEEE
DOI 10.1109/REW53955.2021.00022

II. Industrial Context

Privacy Analytics Inc. (PA for short, hereafter) is an IQVIA company specializing in data privacy. IQVIA is a large multinational company that uses data and advanced analytics to help drive healthcare forward. PA provides anonymization software and services that enable organizations to responsibly use their sensitive data assets while, at the same time, complying with the laws and regulations that govern data sharing. PA relies heavily on agile development. More specifically, PA's practices are based on the Scrum Agile methodology by Schwaber and Beedle [17]. PA sponsored this research in the spirit of driving continuous business process improvements and to contribute lessons learned to the broader software industry and agile practices. Currently, PA has a 27-member engineering team organized into two approximately equally sized agile sub-teams. These two sub-teams are usually split further into smaller break-out teams based on needs at any given point in time.

The entire team works on two-week sprints and participates in story authoring. Most of the stories are product features decided by product managers; the remainder originate from a diverse set of sources, including data scientists, customer support, documentation experts, and engineers. The style, level of detail, and content of the stories can vary.

At the beginning of every sprint, the whole team typically looks at the backlog that has been refined and prioritized beforehand by the product owner. The team then uses Planning Poker [7] to estimate stories that have no story points assigned. This works as follows: every team member picks a card from a deck that represents story points in a Fibonacci series which is widely used in effort estimation [21]. Using a baseline story that has been chosen for each Fibonacci element up to 13 (our maximum story points in a two-week sprint), each member compares the story at hand with those baseline stories to assign a comparable story point.

After all members pick their estimate, the team reveals their cards. Estimation for the given story is concluded if the whole team's estimates fall within three consecutive Fibonacci numbers from one another. For example, if there are four members from the team present at the estimation meeting, and the cards picked are 1, 2, 2 and 3, then the Scrum Master assigns the average, 2 points. However, if 3, 5, 8 and 13 were picked, then the outliers (3 and 13 in this case) would have to explain the rationale behind their estimation. Once any confusion has been cleared up, the team re-votes. If the new votes are still not within three consecutive Fibonacci numbers, the average is assigned to the story and the team moves on to the next story. While PA strives to follow best practices, points are not always assigned in the same manner. In some cases, stories are assigned points after the implementation is complete. Other stories are estimated by the developer who will work on the implementation. The stories are all stored in Jira. At the office, team members use a Planning Poker deck of cards, and virtually, the team uses the Scrumpy Poker tool (https://scrumpy.poker). We use the existing (human-provided) story points as a gold standard for training and evaluation, as we discuss in Sections IV and V.

PA's agile practices provide an ideal setting for our study. The fact that estimations are not consistently performed beforehand and not by the same people puts forward a typical environment in the industry.

III. Related Work

There are several papers in the literature that propose a machine learning-based solution to story-point estimation. Chongpakdee et al [4] use document fingerprints – a technique traditionally used for detecting plagiarism – to search for similar stories and use their points in estimating new stories. This approach combines Hypergeometric Language Model (HLM) and Random N-gram Sample (RNS) techniques to produce a similarity score between the issues from the training set and the issue of interest from the testing set. The authors evaluate their approach using Mean Magnitude of Relative Error (MMRE), which is the average ratio of the absolute error, i.e., the absolute difference between predicted and actual estimations. The evaluation is broken down by issue type (such as bug, story, change request). Their dataset includes 12 open-source projects, achieving an average MMRE of 1.135 with the "change request" issue type scoring the best MMRE performance at 0.26.

Scott and Pfahl [19] compare two Support Vector Machine (SVM) models: one that consumes developers' features as input and another that consumes textual features. The developers' features included are the developer's reputation, workload, total number of points and stories completed and the number of comments left on issues. The textual features include the summary and description of issues, the number of characters, and the n-grams used to calculate Term Frequency-Inverse Document Frequency (TF-IDF). The authors perform an evaluation on eight open-source projects, with the developers' features-based model outperforming the baseline in only three out of eight projects according to Mean Absolute Error (MAE). The paper concludes that a model using developers' features as input can perform up to 10% better on average than only textual features.

Porru et al [16] propose another SVM-based solution. They include one industrial project alongside eight open-source ones in their evaluation. The authors isolate code from the stories and extract its features separately, since code snippets are semantically different from natural-language sentences. Here, story-point estimation is cast as a classification problem, where the model predicts from a multi-class set of labels mapped to the Fibonacci sequence. The model achieves 29% better performance than the closest baseline on their industrial project with an average accuracy of 64% across projects. By comparison, our dataset is nearly seven times larger than Porru et al's industrial project. We experiment with a different estimation technique (Deep-SE, discussed next) and provide an in-depth examination of the characteristics of our dataset, thanks to the first author's familiarity and expertise with the dataset.

As briefly explained earlier, Choetkiertikul et al. [3] introduce a deep-learning solution, named Deep-SE. This solution does not require manual feature extraction. Deep-SE, the tech-

nical platform for our experimentation in this paper, is built on a Long Short-Term Memory (LSTM) architecture [8]. The model generates pretrained word embeddings from stories' summaries and descriptions to feed a Recurrent Highway Network where a regression function predicts the story points. The authors include 16 open-source projects in their dataset from nine different repositories. The accuracy achieved is not consistent across projects, spanning a wide range of MAE from 0.64 to 5.97. In Section V, we investigate how accurate Deep-SE is on our commercial dataset.

IV. Our Adaptation of Deep-SE

Our approach for predicting story points and evaluating machine learning model performance is a close adaptation of Deep-SE. Below, we briefly review the Deep-SE approach, followed by a discussion of the new elements we introduce in our evaluation. For further details about Deep-SE, consult [3]. Deep-SE can be summarized in the following three steps:

1) **Pretraining:** Word embeddings are derived by feeding an *unlabeled* dataset of stories to an LSTM network. The LSTM model's goal in the pretraining step is to predict the next word.
2) **Training:** Using a *labeled* dataset, the summary and description of each story are concatenated and fed to another LSTM network in order to produce a single vector representing the story. The resulting vector representations are then fed to a Recurrent Highway Network (RHN). The story-point predictions (output) are generated by this network's final layer, which is a linear regression activation function. We explain our training, validation and test datasets in Section V-B.
3) **Evaluation:** Model performance is measured against the same baselines used in [3]: mean, median and random guess.

To explore how the level of detail in stories correlates with model performance, we analyze the breakdown of model performance by story length. This analysis is new in our study and was not done by Choetkiertikul et al. [3] for their datasets.

Different story types have different templates. For example, a defect will have expected behaviour, observed behaviour and possibly stack traces while feature requests often include minimal details as they are created before being reviewed by a technical-team member. Therefore, we introduce in our work an analysis of the model performance by story types such as defects and feature requests. Furthermore, in an effort to examine possible correlations between low-quality stories and model performance, we collect the most frequent words and bi-grams that appear in low-performing stories to detect any patterns and compare them against Arora et al. [1]'s list of vague words in natural-language requirements. Low performance is defined as stories with absolute error greater than 1.5. Vague-word analysis is new in our work and was not explored by Choetkiertikul et al. [3].

Following Choetkiertikul et al. [3], we measure the statistical significance of our results against the baselines using Wilcoxon Signed Rank statistical significance test [12] to determine whether model predictions are better than chance. Also similarly to Deep-SE, we measure the likelihood that a model trained on our dataset will perform better than the baselines by calculating the effect size using Vargha and Delaney's $\hat{A}_{XY}$ statistic [23]. This can help establish a threshold of what constitutes a "good-performance model" to use in story-point estimation. Finally, we compare the model performance and our dataset summary statistics against Choetkiertikul et al.'s results over open-source projects; the goal here is gain preliminary insights about whether industrial projects possess characteristics that may not exist in open-source projects and whether any such characteristics can (potentially) impact the use of machine learning for story-point estimation.

V. Empirical Evaluation

In this section, we evaluate Deep-SE on PA's dataset.

A. Research Questions

Our evaluation aims to answer the following two research questions (RQs):

- **RQ1: Performance.** *How well does Deep-SE predict story points over PA's dataset?* Our dataset is discussed in Section V-B. The metrics and baselines used in our evaluation are discussed in Section V-C.
- **RQ2: Characteristics of stories.** *What characteristics do the stories in PA's dataset possess that correlate with the performance of machine learning-based story-point estimation?* To answer RQ2, we perform a number of data exploration operations on PA's dataset, as we explain in Section V-C. We note that our data exploration is not meant at establishing causality between the characteristics of stories and the accuracy of machine learning-based estimation. Instead, we focus on making early observations and identifying interesting correlations that help pave the way for future investigations of causality.

B. Dataset Description

Our dataset is a collection of stories authored in Jira since its adoption at PA in January 2014 until October 2020, when the work for this paper started. The data was split into two mutually exclusive subsets: one for the generation of word embeddings, which we will refer to as the pretraining dataset, and the other for generating and evaluating the fitted model, which we will refer to as the gold-standard dataset. We discuss the selection criteria for each dataset in its respective section below.

a) Pretraining dataset: The pretraining dataset consists of 11,306 stories that do not contain story points. Since during pretraining the model aims to predict the next word, labels (story points) are not applicable. We extract stories from five internal projects and combine them into one dataset. Since the same engineers often work on different projects and since the different break-out teams often have the same Scrum Master at PA, we do not split the dataset by team. While some teams use specific terms in authoring their stories, all teams work on the same anonymization product and thus share the vast

TABLE I
SUMMARY STATISTICS FOR PRETRAINING DATASET

Token Length	Mean	122
	Median	75
	Minimum	1
	Maximum	5807

majority of internal jargon and terminology. We include stories in the dataset that are either open (to do or in progress) or closed (done) but *without* story points. We do this to ensure the pretraining and gold-standard datasets are mutually exclusive.

The mean, median, minimum and maximum token length of the pretraining dataset is shown in Table I. Since certain Jira-specific syntax, such as linking stories, are exported as non-ASCII characters, the dataset was cleaned to include only ASCII characters. Due to the proprietary nature of the dataset and the confidentially agreement signed, we provide in Table II only a few examples that do not contain sensitive information.

b) Gold-standard (Labeled) Dataset: The gold-standard dataset contains stories that were already labeled with story points, subject to the following exclusions: At PA, a story may have been assigned zero points. This could imply that the story does not require any effort, thus acting only as a reminder. Alternatively, it could be that the story was created but subsequently invalidated or rendered obsolete. We still like to keep such stories for tracking and possibly recycling in the future. Stories that are assigned more than 13 points from the Fibonacci sequence either imply that this is a large effort needing to be broken up further, or that we need additional information before we can perform an estimation. For these reasons, zero- and above-13-point stories are excluded from the gold standard, as these stories do not reflect valid estimations. Finally, we exclude stories that are marked as either "Won't do" or "Duplicate" prior to training to ensure that the gold-standard labels are applied only to feature implementations or bug fixes that were actually delivered.

A few non-sensitive examples from our gold standard are shown in Table III. Notice that not all our stories follow the agile user-story template: "As a *[user]*, I would like to *[function]* so that I can *[benefit]*". Table IV provides summary statistics for the gold standard. These statistics include the same metadata reported by Deep-SE authors, plus metadata related to token length as introduced in Section IV. The hypothesis for including token length is that a high level of detail in a story could clarify uncertainty and narrow the scope, and in this way, make it easier for teams to estimate the associated effort.

Choetkiertikul et al. [3] sort their datasets by creation date and then split the top (oldest) 60% of the stories and the bottom (newest) 40% for training and validation/testing, respectively. The 40% portion is further split evenly into validation and testing sets. In our study, we need to adapt this splitting strategy. The observation that prompts the adaptation is as follows: the distribution of story points in the validation/testing dataset would drift from that of the training set, if we were to apply a strictly chronological 60-40 split. An examination of our dataset determined that this difference in the distributions originates from the newest stories being predominantly estimated at low points (one and two), with the higher story points ($\geq$3) being scarce or absent from the newest stories. This led to a follow-up investigation as to why this phenomenon was happening. One hypothesis that we considered was that story-authoring and estimation practices at PA may have evolved drastically over time. If this turned out to be true, a reasonable remedy for the observed discrepancy would have been to discard some of the oldest stories, on the grounds that these stories would no longer be representative of our current practice. After discussing this hypothesis with the team, we did not find evidence of drastic methodological changes, despite the improvements that had been made to the practices over the years.

Our investigation instead led to a different explanation for the phenomenon observed: there was a natural but implicit progression from more abstract to more concrete stories as our product went through the inception, design and implementation stages. With the ability to be more concrete, the engineers tended to define smaller units of work, being cognizant of the fact that these smaller (and typically more sharply scoped) units bode better with project planning and schedule-risk reduction. We expect this process of going from more abstract to more concrete stories to be *cyclic* and repeating itself as major new components and services are introduced into our product. To gauge the potential for adopting an automated story-point estimation technique, we need to study its performance on the full spectrum of stories that the technique may encounter, ranging from the more abstract (often entailing higher story points) to the more concrete (often entailing lower story points). To achieve this evaluation objective and at the same time maintain the realism of chronologically ordering the stories, we apply the following procedure:

1) We calculate the distribution of story points for our entire dataset, shown in Table V;
2) We rebuild the dataset by chronologically picking stories while maintaining the same distribution across the training and validation/testing sets. For example, the top 60%, i.e., the training set, will contain 47% [1] stories with one story point, and these stories will chronologically be the oldest ones in that category.

C. Analysis Procedure and Metrics

To answer **RQ1**, we evaluate the model on the test portion of the gold standard using the following metrics:

- Mean Absolute Error (MAE): the average of the absolute difference between the actual and predicted story points, shown in Equation 1.

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

[1] See the first row in Table V.

TABLE II
PRETRAINING DATASET SAMPLE

issuekey	title	description	storypoint
CS-32	Support ZIP files with a thread per file	NULL	NULL
CS-149	Anonymize files given a source and target directory	NULL	NULL
CS-5217	User names should not be case sensitive	As a user I want to enter my username without worrying about the case that the admin used when they first created my user account. User names are currently case sensitive. We should disable this so that the case of the username does not matter.	NULL

TABLE III
GOLD-STANDARD DATASET SAMPLE

issuekey	title	description	storypoint
CTT-226	RemoveMarkup - Add better help tips	As a user I would like better descriptions for the RemoveMarkup options when accessing the help file. Definition of Done: Each option is defined with a short phrase defining what it does.	2
ECTN-256	Add specifying Masking functionality	NULL	3
EE-968	User should be able to get the list of available rules set	User should be able to get the list of available rules set using a REST API.	5

TABLE IV
SUMMARY STATISTICS FOR GOLD STANDARD

Measurement	Story Points	Token Length
Minimum	1	2
Maximum	13	6975
Mean	3.10	125.68
Median	2	56
Mode	1	8
Variance	8.99	75419.28
Std	2.99	274.63

TABLE V
STORY-POINT DISTRIBUTION

Story Point	Count	Percentage
1	2205	47%
3	802	17%
5	649	14%
2	465	10%
8	399	8%
13	207	4%

- Median Absolute Error (MdAE): the median of absolute errors: $Median\{|actual - estimated|\}$
- Standardized accuracy (SA) [9]: the measurement of how much better the model performed than random guess MAE_R shown in Equation 2. An SA of 0 means the same as random guess, a positive number means better than random guess, and negative number means worse. The random-guess estimate is (randomly) chosen from the training subset.

$$SA = \left(1 - \frac{MAE}{MAE_R}\right) \times 100 \tag{2}$$

We calculate the above metrics for the three baselines used by Choetkiertikul et al. [3]:

1) Mean estimation: Use the average story points of the training dataset to estimate all stories in the test dataset
2) Median estimation: Use the median story points of the training dataset to estimate all stories in the test dataset
3) Random guessing: For every story in the test dataset, pick a story point randomly from the training dataset to estimate the story at hand

In order to analyze whether the results achieved are better than chance, we calculate Wilcoxon statistical significance test with a confidence limit $p = 0.05$. The null hypothesis is "the performance of Deep-SE model in terms of absolute error is not different from other estimation methods such as mean, median or random estimation".

Finally, to measure the likelihood of Deep-SE model trained on our dataset performing better than the baselines, we calculate the effect size $\hat{A}_{XY}$ using Equation 3 where n is the number of stories in the test dataset and $Count(X < Y)$ is the count of instances where Deep-SE (X) achieved absolute error less than the estimation method baseline (Y). Similarly, $Count(X = Y)$ is the count of instances where Deep-SE achieved an equal absolute error to the baseline Y.

$$\hat{A}_{XY} = \frac{Count(X < Y) + (0.5 \times Count(X = Y))}{n} \tag{3}$$

To answer **RQ2**, we follow the steps described below:

- We perform k-means to cluster word embeddings from the pretrained model. We use the same script included in the Deep-SE package to calculate the clusters. The script uses t-distributed stochastic neighbour embedding [22] to convert high-dimensional vectors to two dimensions, so that the clusters can be more easily visualized. This enables us to assess whether the model can learn the internal jargon that exists in an industrial setting. Similar topics appearing closer together in the clusters indicates the model's understanding of the domain.

- We analyze the performance of stories without a description in comparison to those with a description. We pick the best-performing open-source project reported in [3] and compare the percentage of stories with no description in its dataset to that in our dataset. The hypothesis here is that a story with a description should convey more detail than just a story with a summary. This enables us to draw insights as to whether stories in an industrial context may be missing a description compared to open-source projects, and whether that correlates with model performance.
- In order to examine whether having more details in a given story has a relationship with prediction error, we analyze the correlation between average token length and MAE in our dataset. If there is a correlation, we look at whether the correlation exists in open-source projects as well. This helps establish grounds for examining (in the future) whether a more detailed story translates to better model performance.
- We look for patterns unique to either open-source or our industrial dataset by comparing the average token length in our dataset to the open-source project with the closest performance and the open-source project with the best performance. The reason for not collating all the open-source datasets is that the MAE reported by Choetkiertikul et al. [3] varied widely ranging from 0.64 to 5.97, and therefore, we were interested in comparing characteristics of projects where the model performed similarly.
- To look for commonalities between vague words appearing in open-source versus our commercial dataset, we build a list of most frequent tokens and bi-grams which appeared in the low-performing stories (with $AE > 1.5$) but which did not appear in the high-performing stories. We compare the results according to the vague list of words by Arora et al. [1]. Furthermore, we look at whether these vague words appear more frequently in open-source projects or vice-versa.

D. Results and Discussion

Below, we present and discuss our results, organized by RQ1 and RQ2:

*1) **RQ1 (Performance)**:* Table VI presents Deep-SE performance on our dataset ordered by MAE and SA from best to worst. As shown, Deep-SE performed better than all 3 baselines achieving MAE of 1.46 and SA of 48.45%. Table VII shows the performance broken down by story points. The best results were obtained for stories with 1, 2 and 3 points while the model struggled with story points 5, 8 and 13. On the surface, this can be explained by the fact that the higher the story point, the more impact errors in the model's predictions will have on the average. However, looking at Table VIII, which shows the breakdown of average token length by story point in the test dataset, we can deduce a different insight: Pearson's correlation tests (detailed in Section V-D2) show that stories with high average token length performed better than stories with low average token length. We note that what we have observed is a correlation and not necessarily causation. Further statistical analysis is required to scope the relationship between token length and performance and investigate causality. Another related observation is that performance by story points did not always correspond to the distribution of the story points in the dataset. For example, as shown in Table V, the dataset had less instances of stories with two story points than three, and yet, the model performed better on stories with two points.

TABLE VI
PERFORMANCE OF DEEP-SE AGAINST BASELINES

Method	MAE	MdAE	SA
Deep-SE	1.46	0.78	48.45
Median	2.04	1.0	27.89
Mean	2.22	2.1	21.7
Random	2.83	2.0	0

TABLE VII
DEEP-SE PERFORMANCE BY STORY POINT

Story Point	MAE
1	0.69
2	0.87
3	1.06
5	2.15
8	4.5
13	4.48

TABLE VIII
AVERAGE TOKEN LENGTH BY STORY POINT IN TEST DATASET

Story Point	Average Token Length
1	134.78
2	182.72
3	134.6
5	96.87
8	72.04
13	63.62

Table IX presents the results of the statistical significance tests. As shown, Deep-SE results against all baselines were significantly lower than $p = 0.05$. Therefore, the null hypothesis is rejected. Furthermore, the likelihood of Deep-SE performing better than mean, median and random baselines averaged at 65% where it ranged from 63% for random to 67% for the mean baseline. The effect size of Deep-SE on our dataset was lower than reported by the original Deep-SE paper on open-source projects where it averaged around 77%.

TABLE IX
STATISTICAL SIGNIFICANCE RESULTS

Method	p-value	$\hat{A}_{XY}$
Mean	3.94e-39	0.67
Median	6.57e-29	0.64
Random	3.9e-41	0.63

Deep-SE performed best on Jira issues marked as "Bug" even though they contained non-natural language elements

TABLE X
MAE BY JIRA ISSUE TYPE

Issue Type	MAE
Bug	0.76
Epic	0.77
Support Request	0.78
Task	1.19
Feature Request	1.22
Sub-task	1.69
Story	2.00
Improvement	2.38
New Feature	3.00

TABLE XI
MAE BY INTERNAL PROJECT

Project	MAE	Average Token Length
CS	1.25	154.55
ET	1.28	89.17
ECTN	1.56	79.61
EE	1.59	60.31
MIR	1.86	82.54
CTT	2.26	73.56

such as stack traces and log snippets. The model, on the other hand, performed worst with issues marked as "New Feature". We believe that this is due to the fact that, at PA, a new feature request is typically not vetted by a technical team member and important implementation details can potentially be missing. As we noted in Section II, high points are assigned when there is uncertainty around implementation. This reflects on the 7.75 average story points assigned to "New Feature", 1.82 times higher than the average points assigned to Jira type "Story". Table X shows a breakdown of MAE by issue type sorted from best performance to worst.

Finally, we present in Table XI the MAE broken down by internal PA sub-projects. This is another view of the model's performance on the test portion of our dataset. The results in Table XI also shows the correlation between average token length and the model's MAE score. The project with the lowest MAE also had the highest average token length.

We note that for RQ1, we used the same hyperparameter values used by Choetkiertikul et al. [3], except for maximum sequence length, which Choetkiertikul et al. set to the average token length of their (combined) dataset. We set this hyperparameter according to our dataset, where the average token length is 125. For a complete list of Deep-SE's hyperparameters and their values, see [3].

*2) **RQ2 (Characteristics of stories):*** To develop insights about aspects that might have impacted Deep-SE's performance on our dataset, we first look at whether the pretrained model could learn the semantics of our industrial jargon. Figure 1 shows how Deep-SE clustered the terms that appeared in the gold-standard dataset using the pretrained model's embeddings. We can see that the model grouped terms related to the product together such as product, customer, and license. It also clustered de-identification domain knowledge such as

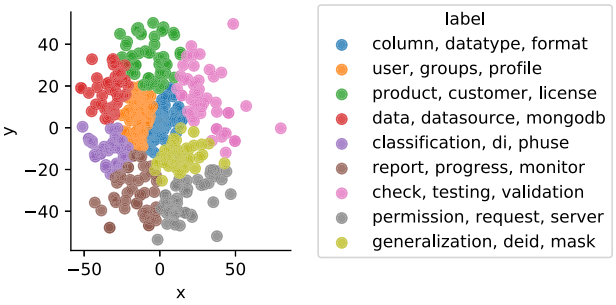

Fig. 1. Semantic Clusters

generalization, deid, mask. It therefore learned the semantics of data anonymization terms and correctly placed them in the same clusters. The model, however, placed terms related to data or databases farther apart. For example data, datasource, mongodb were placed in a different cluster from column, datatype, format. This inconsistent clustering of related terms was also reflected by Deep-SE performance in predicting the next word during pretraining where it only achieved an accuracy of 60%.

To facilitate our comparison with open-source projects reported by [3], we will refer to the projects by their abbreviations used in [3] and introduce a new abbreviation for our dataset:

- **TE**: The project that achieved the best Deep-SE performance, MAE of 0.64
- **JI**: The project that achieved the closest performance to our dataset, MAE of 1.38
- **PA**: Our dataset

It was observed that PA had a higher percentage of issues with no description than TE. PA contains 827 stories with no description which amounts to 17.5% of stories compared with 10% of TE and 18.75% of JI. While this shows that higher percentage of stories with no description is not an attribute of commercial projects alone, it reinforces the observation that a high percentage of stories with no description negatively impacts performance. For example, in PA, the MAE of the stories with no description was 1.88 compared to 1.37 for those with a description. The distribution of stories with no description, however, corresponds to the model's performance by story points. For example, stories with one story point where Deep-SE performed better only had 9.34% of their stories without a description. On the other hand, stories with 13 story points where Deep-SE performed worse had 26.09% of their stories with no description.

This finding also reinforces the observation around the correlation between average token length and MAE briefly introduced in Section V-D1. In Jira, the title has a character limit of 255. Therefore, issues with no description lowered the average token length. In order to validate this relation, Pearson's correlation coefficient was calculated for the PA dataset and the result was a strong negative correlation of -89% between the token length and MAE; the higher the token

TABLE XII
TOKEN-LENGTH COMPARISONS BETWEEN PA AND TE

Measurement	PA	TE
Mean	125.68	122.73
Median	56	73
Std	274.63	198

length, the lower the MAE and thus the better the performance. While the average token length of TE was similar overall to PA's, the standard deviation was higher in PA than TE and the median in PA was lower than TE as shown in Table XII. This suggests that there is a discrepancy between content length in our dataset compared to TE. There was no strong correlation found, however, between token length and MAE in the TE dataset. This suggests that token length is not the only predictor of strong model performance. Practitioners should not assume that merely adding more details in stories will automatically translate to good model performance.

The PA dataset was also analyzed for frequent tokens that appeared only in stories with AE higher than 1.5. The threshold for frequency was set at 100. A total of three frequent tokens matched the vague-word list compiled by [1]: {"also", "support", "process"}. When including synonyms and other verb tenses, the list grew to seven tokens, where synonyms of "next", "use" and "assure" appeared as "following", "using", "ensure" and the bi-gram "make sure". Each of these words appeared more than 100 times in high-AE stories. In contrast, while low-AE stories make up 65% of the test dataset and thus the odds of vague words appearing in them are 1.8 times as high, no vague words above our frequency threshold were observed in the low-AE stories. We would, however, caution the reader not to deduce that the existence of vague words directly impacts the model's performance: a definitive link could not be drawn between the existence of those words and the model's predictions. Further explainability analysis is required to corroborate or rule out possible correlations between vague words and a machine learning model's story-point predictions.

The other bi-grams associated with high MAE included "acceptance criteria", "implementation details" and "hit continue". This may appear as a contradiction to the previous result that more content lowers MAE. While these bi-grams imply more detailed information, we note that, since the maximum sequence length was set to the average of 125 tokens, the extra tokens did not play a role in the model's performance. Due to the high standard deviation in token length in the PA dataset, increasing the maximum sequence length did not impact overall performance. This shows that consistency around the level of detail reflected in low standard deviation is what matters and not just the average token length.

In comparison to the TE and JI open-source projects, we observe that only one vague word appeared in JI's high-AE stories and two vague words in TE's. It is worth noting though that PA's dataset is approximately four times larger than both these projects combined.

E. Threats to Validity

Below, we discuss the steps we took to mitigate the most pertinent threats to the validity of our empirical analysis.

1) Internal Validity: Our dataset is comprised of nearly six years worth of stories for a commercial product. The data was not manipulated in any way to avoid biasing the results. Using the dataset as-is helped ensure that the results are reflective of a real context. While the stories and their story-point estimates may not be fully consistent or accurate and may have been produced by experts with their own biases, this is aligned with the real world in an agile software development team.

2) Conclusion validity: To increase conclusion validity, several statistical significance tests were performed to ensure there is a statistical relationship between the analysis and the conclusion. Wilcoxon Signed Rank Test with p-value significantly lower than 0.05 was confirmed. Vargha & Delaney's effect size test was also performed against all three baselines. The conclusions were also verified by comparing the results to the open-source projects used by Choetkiertikul et al. [3].

3) External validity: As with virtually any other single case study, we cannot argue with confidence about the generalizability of our findings. This being acknowledged, an effort was made to include in our analysis as many potentially relevant factors as possible (e.g., token length, percentage of stories with no description, and presence of vague words). Since our dataset is proprietary and cannot be shared, we provide metadata (summary statistics) for these factors to depict as precise a characterization as we can of our dataset. This allows others to compare the metadata for their dataset against ours, and in this way, determine whether our study is representative of their application domain as well.

VI. LESSONS LEARNED

In this section, we present the lessons learned from our investigation. The lessons are primarily intended at helping practitioners who are considering the use of automated methods such as machine learning for estimating story points.

A. Accuracy of ML Predictions

Choetkiertikul et al. [3] formulate story-point estimation as a regression problem in order to broaden applicability to a wide range of agile practices. We believe that, in many practical situations, using classification, as opposed to regression, will suffice, with the added benefit that classification-based estimation could be more accurate. The intuition here is as follows: In an industrial context, the range $\{1, 2, 3, \ldots, n\}$, where n is the maximum number of story points in a given sprint, is typically mapped to classes such as {Extra-Small, Small, Medium, Large, Extra-Large}, also known as affinity estimation or T-shirt sizing estimation.

We believe that story-point estimation is not expected to be as accurate as other regression problems such as weather forecasting. For example, if 1-3 points is mapped to small effort and a regressor predicts a 1-point story to be a 3-point story, it will have an absolute error of 2, which could be considered poor performance. However, if a classifier predicts

the same story to be of small effort, accuracy will be 100%. When a Fibonacci sequence is used, the performance discrepancy between a regressor and a classifier becomes even more pronounced for the higher end of story points. For example, 8 story points predicted as 13 would cause an absolute error of 5, while a classifier could accurately predict it as large effort. In light of the above, practitioners need to interpret the mean absolute error of a regressor with care; a seemingly large mean error is not necessarily a sign that machine learning is a poor fit. If an organization is using the T-shirt sizing method, we recommend that they either map the regressor's predictions to discrete (ordinal) values, or alternatively, opt for classification right from the outset.

B. Quality of Existing Estimates

At PA, developers do not typically go back to modify an estimation based on actual effort after development is complete. This can cause the estimates to not qualify as a "gold standard" for actual effort. While Planning Poker aims to debias estimates, we observed that in practice there is the possibility that there was a "go-to" story point during estimations or that anchoring was an issue, with one developer potentially biasing the rest of the team's estimates. Naturally, the quality of (supervised) machine learning can be only as good as the quality of the labeled data that is available. In our context, this means that agile teams need to closely examine their estimation practices and determine whether the quality of their existing estimates is high enough to support the construction of accurate prediction models. To this end, agile training can help solidify a team's application of expert estimation. Holding regular retrospectives every iteration to self-assess a team's adherence to agile guidelines can help continuously improve story authoring and story-point estimation.

C. Level of Detail in Stories

The high correlation observed between MAE and average token length warrants further investigation as to whether the level of detail provided in stories directly impacts a machine learning model's performance. While varying levels of details are to be expected as discussed in Section II, the backlog can be broken up further by story type such as defects, technical debt, product features and reminder tasks. An important message for practitioners here is to try to minimize tacit information in their stories. For example, it is common to create a story to remind the assignee that they need to work on a specific task. This task is sometimes the result of side conversations or ad-hoc design meetings, and the story creator may consider that adding more details is redundant, since the developer assigned already knows what is required. If a considerable number of stories in an organization are written this way, one cannot expect meaningful automation to emerge from analyzing the content of stories. To mitigate this problem, we recommend regular backlog grooming sessions between the product manager and the Scrum Master to add missing details to stories. Developing a template that the whole team adopts, e.g., Story Goal, Acceptance Criteria, Design Considerations and Uncertainties, can help write more detailed stories.

D. Reducing Uncertainty and Vagueness

It is known that uncertainty in requirements impedes estimatability [11]. Our results provide preliminary empirical evidence for this phenomenon in the context of story-point estimation. In particular, we observed that three vague terms, and seven if we include their synonyms, appeared frequently only in stories that have high estimation error. Agile teams should strive to clear up any uncertainties before an iteration starts. In particular, we recommend that analysts check for the presence of vague terms and requirement smells [11] in their stories. Authoring guidelines may be needed to help minimize the common pitfalls associated with the use of natural language in stories. While unanticipated issues may inevitably arise during implementation, we believe team leads can reduce such issues by ensuring that uncertainty and vagueness in stories is reduced as much as possible, e.g, by holding review sessions prior to the implementation sprint.

E. Prerequisites for Adoption

We presented the results of this research to the team at PA and asked the team what prerequisites they thought would need to be met before a machine learning approach could be adopted as either an enhancement to or a replacement for the current manual method based on Planning Poker. Briefly, grooming a backlog of high-quality stories with actual effort adjusted after development completion was noted as the main prerequisite for future adoption.

In relation to the quality of stories, the team was of the overall opinion that improvements would present numerous benefits, beyond supporting automated estimation. As for being more methodical about recording the actual effort after story completion, the team felt that, given PA's current workflows, recording this information can be accommodated without major overhead. The team nonetheless noted that it would likely take months if not years for enough high-quality data to accumulate before machine learning can be applied reliably. Furthermore, the team felt that, to be able to extrapolate from historical data, the data needs to also cover various factors that can influence the estimates at any given point in time (e.g., development platform, deployment environment, and applicable regulations). A more holistic data collection strategy may thus be required. Whether such a strategy can be reconciled with agile principles is something that requires further investigation.

VII. Conclusion

We evaluated an existing state-of-the-art machine learning technique, Deep-SE [3], for estimating story points in a large commercial product developed using agile methods. By doing so, we derived insights about the applicability of machine learning-based story-point estimation in a non-open-source setting, noting that Deep-SE had previously been evaluated on open-source projects only. We further examined some simple

but important textual characteristics of non-open-source stories versus open-source stories and studied in a systematic way how these characteristics could relate to the performance of automated story-point estimation.

The Deep-SE model that we built over our commercial dataset significantly outperformed three different baselines. This provides initial but useful evidence about the applicability of Deep-SE outside the open-source community. The model could also relatively successfully learn the semantics of the industrial jargon that did not appear in the public domain. Our work highlighted several factors that have the potential to negatively affect the usefulness of automated story-point estimation. These include inaccurate historical estimates, the lack of detail in stories, and the presence of vague words in them. Based on the experience gained from our study, we believe that organizations may need to invest time into honing their agile practices and mitigating these factors first, before adopting automated approaches such as machine learning for story-point estimation.

In the future, we plan to build a classification-based model to support affinity estimation. Furthermore, we would like to repeat our study after a period of time and once we have put into practice the lessons that we learned from our current work. We also plan to explore Explainable AI techniques such as integrated gradients [20] to measure the attributions of vague words and requirements smells to the model's predictions.

ACKNOWLEDGMENT

We are grateful to Khaldoun Zine El Abidine and Guillaume Senneville at Privacy Analytics for facilitating and supporting this research. We thank the engineering team at Privacy Analytics for their valuable feedback. The second author is supported by NSERC of Canada under the Discovery and Discovery Accelerator programs.

REFERENCES

[1] C. Arora, M. Sabetzadeh, L. Briand, and F. Zimmer. Automated checking of conformance to requirements templates using natural language processing. *IEEE TSE*, 41(10), 2015.

[2] F. Chatziasimidis and I. Stamelos. Data collection and analysis of github repositories and users. In *IEEE IISA'15*, 2015.

[3] M. Choetkiertikul, H. Dam, T. Tran, T. Pham, A. Ghose, and T. Menzies. A deep learning model for estimating story points. *IEEE TSE*, 45(7):637–656, 2019.

[4] P. Chongpakdee and W. Vatanawood. Estimating user story points using document fingerprints. In *IEEE ICSESS'17*, pages 149–152, 2017.

[5] Mike Cohn. *Agile estimating and planning*. Prentice Hall, 2012.

[6] R. Goswami, M. Jasuja, and S. Dhir. Impact of different estimation approaches in traditional and agile development. In *IEEE CICT'16*, pages 679–682, 2016.

[7] James Grenning. Planning poker or how to avoid analysis paralysis while release planning. *Hawthorn Woods: Renaissance Software Consulting*, 3, 2002.

[8] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Computation*, 9(8):1735–1780, 1997.

[9] Ali Idri, Ibtissam Abnane, and Alain Abran. Evaluating pred (p) and standardized accuracy criteria in software development effort estimation. *Journal of Software: Evolution and Process*, 30(4), 2018.

[10] N. Keshta and Y. Morgan. Comparison between traditional plan-based and agile software processes according to team size project domain (a systematic literature review). In *IEEE IEMCON'17*, 2017.

[11] Garm Lucassen, Fabiano Dalpiaz, Jan Martijn van der Werf, and Sjaak Brinkkemper. Improving agile requirements: the quality user story framework and tool. *REJ*, 21(3), 2016.

[12] Keith Muller. Statistical power analysis for the behavioral sciences. *Technometrics*, 31(4), 1989.

[13] M. Müller. Agile challenges and chances for open source: Lessons learned from managing a floss project. In *IEEE ICOS'18*, 2018.

[14] M. Owais and R. Ramakishore. Effort, duration and cost estimation in agile software development. In *IEEE IC3'16*, pages 1–5. IEEE, 2016.

[15] R. Popli and N. Chauahn. Managing uncertainty of story-points in agile software. In *IEEE INDIACom'15*, pages 1357–1361, 2015.

[16] Simone Porru, Alessandro Murgia, Serge Demeyer, Michele Marchesi, and Roberto Tonelli. Estimating story points from issue reports. In *ACM PROMISE'16*, 2016.

[17] Ken Schwaber and Mike Beedle. *Agile Software Development with Scrum*. Prentice Hall, 2001.

[18] E. Scott, K. N. Charkie, and D. Pfahl. Productivity, turnover, and team stability of agile teams in open-source software projects. In *IEEE SEAA'20*, 2020.

[19] Ezequiel Scott and Dietmar Pfahl. Using developers' features to estimate story points. In *ACM ICSSP'18*, 2018.

[20] Mukund Sundararajan, Ankur Taly, and Qiqi Yan. Axiomatic attribution for deep networks. In *ICML'17*, 2017.

[21] R. Tamrakar and M. Jørgensen. Does the use of Fibonacci numbers in planning poker affect effort estimates? In *IET EASE'12*, 2012.

[22] L.J.P. van der Maaten and G.E. Hinton. Visualizing high-dimensional data using t-sne. *Machine Learning Research*, 9, 2008.

[23] András Vargha and Harold D. Delaney. A critique and improvement of the cl common language effect size statistics of mcgraw and wong. *Educational and Behavioral Statistics*, 25(2), 2000.