On Identifying Similar User Stories to Support Agile Estimation based on Historical Data

Aleksander Grzegorz Duszkiewicz^{1,2}, Jacob Glumby Sørensen¹, Niclas Johansen², Henry Edison¹ and Thiago Rocha Silva¹

¹The Maersk Mc-Kinney Moller Institute, University of Southern Denmark, Campusvej 55, Odense M, 5230, Denmark ²Morningtrain ApS, Rugårdsvej 55A 1 tv, Odense C, 5000, Denmark

Abstract

Accurate and reliable effort and cost estimation is still a challenging activity for agile teams. It is argued that leveraging historical data regarding the actual time spent on similar past projects could be very helpful to support such an activity before companies can embark upon a new project. In this position paper, we discuss our initial efforts towards a software tool that retrieves similar user stories from a database of past projects to support developers estimating effort and cost of developing similar new projects. In close collaboration with a Danish software development company, we developed a tool that employs Natural Language Processing (NLP) algorithms to find past similar user stories and retrieve the actual time spent on them. Developers can then make use of the actual implementation time of these stories as the new estimate or use the value to support their decision on a different estimate. Results of a preliminary evaluation regarding the performance of such a tool in terms of precision and recall showed that it is capable of finding similar stories fairly well, but also showed that different phrasings and wordings can have a high impact on similarity scores.

Keywords

User Stories, Agile Estimation, Natural Language Processing

1. Introduction

Effort and cost estimation is still one of the most critical aspects of agile software development. A fair amount of developers' time is spent on understanding, discussing, and trying to find a consensus on an accurate estimate for user stories. Such an activity is even costlier for companies that need to budget a software project before even gaining a software development contract. Besides that, estimates can easily be misjudged and the development taking longer than first anticipated, which can lead to project loss.

Leveraging historical data from similar past projects has been seen as a good strategy to overcome some of these challenges. Having access to the actual time spent to develop user stories in the past can be very helpful to support developers estimating more accurately the effort required to develop similar user stories at present. In practice, however, it is not trivial to

⁽N. Johansen); hedis@mmmi.sdu.dk (H. Edison); thiago@mmmi.sdu.dk (T. R. Silva)

https://www.sdu.dk/staff/hedis (H. Edison); https://www.sdu.dk/staff/trsi (T.R. Silva)
0000-0001-8961-4663 (T.R. Silva)

^{© 0 2022} Copyright for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

CEUR Workshop Proceedings (CEUR-WS.org)

identify such similar stories from past projects and many agile teams simply do not leverage this asset. Despite fruitful previous research on Natural Language Processing (NLP) to support various activities on requirements engineering more broadly [1], and on user stories more specifically [2], there has been limited work on models and tools applying NLP to identify user story similarity in order to support the agile estimation process [3]. When it comes to commercial agile estimation tools, the closest one to support this process is DoWork.ai¹, which leverages artificial intelligence to find similar tasks. When creating a new task, the tool conducts a search for related historical ones and signalizes how similar they are. It is not clear, however, which kind of NLP model they employ and how such a model has been trained.

In this position paper, we discuss our ongoing efforts towards the development of a software tool to support this process. The tool has been co-designed with a Danish software company following design science principles. It employs NLP to identify and retrieve similar user stories from past projects and suggests three levels of estimates (most-likely, pessimistic, and optimistic) for the new user stories being estimated. Developers can then decide to adopt one of the three suggested estimates, or leverage this information to support the decision upon a new estimate based on the particularities of the project at hand. We report our findings from a preliminary evaluation analysing the current performance of the tool to retrieve similar user stories based on text similarity. In the following sections, we briefly introduce the main elements of the tool and discuss both the preliminary results obtained so far and our next steps on this research.

2. The Proposed Software Tool

The proposed tool has been co-designed with Morningtrain², a Danish medium-sized full service digital agency that specializes in the fields of web design, programming, and online marketing. The company currently employs a weighted three-point estimation technique based on PERT [4] and would like to keep using it.

The tool has been developed using React as the front-end framework and Laravel as the back-end framework. The tool is a standalone system that connects to Jira using OAuth 2.0 and Webhooks, providing a two-way communication of project data. Other technologies used include MySQL, Redis, and the library InertiaJS - SPA.

Our initial implementation strategy to find similar user stories was based on tagging. In order to lower the amount of stories for the system to compare, we first implemented a tag system with the purpose of excluding stories with non-similar tags. Only stories that passed this first tag filtering would be submitted to further NLP analysis. The rationale was that this could help the performance as the dataset grows. After experimenting with some data, however, we decided to abandon this strategy and submit all the stories in the dataset to NLP analysis since we did not notice any relevant performance issue.

When performing NLP, we employed semantic analysis to score semantic similarity among user stories, as literal analysis of text did not produce good results. We implemented semantic similarity analysis using spaCy³. spaCy compares two strings through its similarity function

¹https://dowork.ai

²https://morningtrain.dk

³https://spacy.io

that uses cosine similarity to calculate a similarity score. While spaCy comes with a selection of standard models that can be directly used, when tested with our database, the similarity scores were mostly in the high end regardless of the actual similarity. Faced with this issue, we started looking for a model that could better score semantic similarities of full-text sentences. This made us opting for USE [5], a pre-trained model developed by Google. An open-source implementation of USE [6] made it possible to use the model together with spaCy. USE was trained with a large variety of text and uses high-dimensional vectors which makes it a good choice for general purpose tasks, including text classification, semantic similarity, and natural language tasks. During our tests, the similarity scores with USE were closer to what we were expecting based on a manual analysis of the stories in the database.

The tool compares a new story against all other stories marked with the status of "done" in the database. During the comparison process, the user stories are pre-processed in order to remove the template skeleton (i.e., the sub-strings "*As a*", "*I want*", and "*so that*") since we observed it improves the parsing. The tool currently employs a similarity threshold of 60%, i.e., any user story with a similarity score of 60% (0.6) or above is returned to the user. We use the actual time spent on the highest scoring user story returned as the most likely estimate for the new similar user story being estimated.



Figure 1: Similarity search results.

Figure 1 depicts the user interface where the suggested estimates are presented to the user. In the example, two user stories were found similar to the one being specified. The highest scoring story has a 93% similarity to the one entered. The time spent on this story is used as the most likely estimate for the new user story being estimated. Following the three-point estimation technique employed by the company, the user is also presented with an optimistic and a pessimistic estimate. The optimistic and pessimistic values are calculated based on offset values entered into the system options. The user can choose to manually overwrite the estimates or modify the story's complexity level, which automatically changes the optimistic and pessimistic values. The company's rationale for using the threepoint estimate is that these three values can provide a price range that a customer can expect a story to cost. This has significant business value for the company.

The user can also have access to the full list of similar stories (see Figure 2). From there, s/he can access details

such as match percentage, title of the story, the user story description, and the time spent on the implementation of the user story in the past. These data can be used to check if there are any other stories that are actually more similar according to human reasoning. The user can additionally choose to select a specific story from the list of similar ones and access all the details of that story such as an in-depth story description, technical description, user acceptance criteria, tests, and original estimates. The user can also choose to replace the most likely estimate for the new user story by the actual implementation time of any of the returned user stories.

Similar stories									
МАТСН	TITLE	USER STORY	ACTUAL TIME						
93%	Jump to a section	As a user i want to jump to a section that is relevant to me so that i can read what is relevant for me	1	Story details	Use time				
90%	quickly go to a section that is relevant to me	As a user i want to quickly go to a section that is relevant to me so that i can read what is relevant for me	2	Story details	Use time				

Figure 2: Similar stories modal that shows user story details.

Table 1

Tool's performance in the preliminary evaluation. **TP** refers to True Positives, **FP** refers to False Positives, **FN** refers to False Negatives, and **TN** refers to True Negatives.

Pairs (total)	ТР	FP	FN	ΤN	Precision	Recall	F-measure
169	21	0	4	144	1	0.84	0.91

This feature aims to give the user the possibility of making an informed choice. Two user stories can be very similar, but bear different system-dependent challenges, which can have a deep impact on the time needed to implement the same feature.

Finally, we also implemented a Jira integration feature through which a project with all its underlying requirements and stories can be transferred to Jira. A two-way communication is established through Webhooks, i.e., once a project is exported, besides systematically uploading all local changes to Jira, we also listen for event changes within the project on Jira and reflect those changes back to the tool in order to always have the latest updates in our database.

3. Preliminary Evaluation

We performed a preliminary evaluation to determine the accuracy of USE to identify similar user stories into the tool. The test setup included a database of five user stories from past projects in the company and eight additional stories that were either phrased differently but semantically similar to the original ones, or not semantically similar but using phrasings from the original ones. Each user story was then manually compared to the others by one of the researchers (169 combinations), and then independently reviewed by another researcher. A score of either high or low similarity was recorded for each set. We started the study with a similarity threshold of 70% as we were aiming at closely related user stories. Each pair of stories was subsequently tested using our tool and the results compared to the manual analysis. The results were additionally cross-checked by two researchers.

Table 1 summarizes the results of the evaluation. During the test, no false positives were found. There were, however, a few cases that got closer to the 70% threshold. For example, the two following user stories scored a similarity of 62%:

"As a customer, I want to add a product to my wish list, so that I can find the product next time I visit the page"

"As a customer, I want to remove a product from my wish list, so that it does not appear on the list when I visit the page"

These are sentences that do bear some resemblance but do not include the same operations from an implementation perspective. The sentences are therefore not similar in this context. This suggests that the algorithm is indifferent to the opposite verbs "add" and "remove", or at least that it does not significantly penalize these opposites. This is further underlined by the last phrase in each user story, where in the first user story one should be able to find the product, while in the second one it is implied that one should not be able to find the product on the list.

Conversely, four false negatives were found, all within the 60-70% range. An example is presented with the two following user stories that scored a similarity of 65%:

"As an administrator, I want to search for a specific student, so that I can find the student records"

"As a student, I want to look up my student records, so that I can get a look at the details"

The two user stories describe very similar features but phrased differently. While 65% is a good score, it also suggests that there can be some pitfalls when using different words to describe what is likely the same thing. When testing the tool, we found that replacing a context-giving keyword in one of the sentences with a similar word can have significant impact on the overall sentence similarity score. Our tests showed that similar words that scored low when tested against each other in isolation also lowered the similarity score when used in sentences. As an example, "records" and "details" are probably two different ways of referring to students' personal data in this context. The similarity test of these two words alone returned a low similarity score of 32%, which in turn lowered the overall similarity score of the two sentences. This issue relates more broadly to near-synonymy in user stories, which is a topic investigated by Dalpiaz et al. [7].

As the system was designed to be a supporting tool for the user to make an informed decision, after this preliminary evaluation, we decided to lower the tool's similarity threshold from 70% to 60% before moving to further evaluation. This has allowed for broadening the spectrum of stories returned. It means that, due to the complexity of semantic analysis, a manual process must still be done by the developers to select the most fitting stories for their specific context.

4. Conclusion and Next Steps

In this position paper, we briefly introduced a software tool that employs NLP to retrieve information from past user stories in order to support developers through the agile estimation

process. After the preliminary evaluation discussed in the paper, we hypothesized whether it would be valuable to train a model specifically for the purpose of finding similarities between user stories. We did not do this originally due to the very limited data available to perform training on. USE has been trained with data from multiple datasets, including the Stanford Natural Language Inference (SNLI) corpus [8] which consists of 570,000 sentence pairs. Another possibility to achieve better results would be replacing USE by other pre-trained models such as the InferSent model released by Facebook [9], which is also a general purpose model.

Our next steps on this research include testing the use of different models and/or training our own. We have also planned an evaluation in a real setting, with developers from the company using the tool to estimate user stories for a new project. This would provide valuable insights regarding the role that this kind of tool may play on agile projects in industry. Another interesting future direction would be automating NLP routines to group very similar user stories, but with different implementation times, and calculate a mean implementation time that could be presented to the developer. Another NLP task that could be implemented is to scan the database for possible keywords to be added to a tag bank that can be used in searches.

References

- L. Zhao, W. Alhoshan, A. Ferrari, K. J. Letsholo, M. A. Ajagbe, E.-V. Chioasca, R. T. Batista-Navarro, Natural language processing (nlp) for requirements engineering: A systematic mapping study, ACM Computing Surveys 54 (2021).
- [2] I. K. Raharjana, D. Siahaan, C. Fatichah, User stories and natural language processing: A systematic literature review, IEEE Access 9 (2021) 53811–53826.
- [3] B. Alsaadi, K. Saeedi, Data-driven effort estimation techniques of agile user stories: a systematic literature review, Artificial Intelligence Review (2022) 1–32.
- [4] R. W. Miller, Schedule, cost, and profit control with pert: A comprehensive guide for program management (1963).
- [5] D. Cer, Y. Yang, S. Kong, N. Hua, N. Limtiaco, R. S. John, N. Constant, M. Guajardo-Cespedes, S. Yuan, C. Tar, Y. Sung, B. Strope, R. Kurzweil, Universal sentence encoder (2018). arXiv:1803.11175.
- [6] M. Mensio, Spacy universal sentence encoder, 2021. URL: https://github.com/ MartinoMensio/spacy-universal-sentence-encoder.
- [7] F. Dalpiaz, I. Van Der Schalk, S. Brinkkemper, F. B. Aydemir, G. Lucassen, Detecting terminological ambiguity in user stories: Tool and experimentation, Information and Software Technology 110 (2019) 3–16.
- [8] S. R. Bowman, G. Angeli, C. Potts, C. D. Manning, A large annotated corpus for learning natural language inference, in: Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP), Association for Computational Linguistics, 2015.
- [9] A. Conneau, D. Kiela, H. Schwenk, L. Barrault, A. Bordes, Supervised learning of universal sentence representations from natural language inference data (2017). arXiv:1705.02364.