Hybrid Labels are the New Measure!

Maleknaz Nayebi, Shaikh Jeeshan Kabeer, Guenther Ruhe, Chris Carlson, Francis Chew

Abstract—Developing "Minimum viable products" (MVPs) is critical for start-up companies to hit the market fast and with an accepted level of performance. The (US) Food and Drug Administration FDA mandates additional non-functional requirements in health care systems, meaning that the MVP should provide the best availability, privacy, and security. This critical demand evokes companies to further rely on analytics to optimize the development process. In a collaborative project with BrightSquid, we provide a decision support system based on analogical reasoning to assist in effort estimation, scoping and assignment of change requests. In this experience report, we propose a new metric, change request labels, for better prediction. Using different methods for textual similarity analysis, we found that combination of machine learning techniques with experts manually added labels has the highest prediction accuracy. Better prediction of change impacts allows the company is optimizing their resources and provide proper timing of releases to target MVPs.

Index Terms—Software analytics, Digital health, Digital care, Label, Tag, Change impact analysis

1 Context: Minimum viable product development in Digital Health Care

Brightsquid Secure Communication Corp is a global provider of HIPAA-compliant (Health Insurance Portability and Accountability Act) communication solutions - providing compliant messaging, email, and large file transfer for medical and dental professionals since 2009. Secure health exchange is Brightsquid's core communication and collaboration platform called SecureMail. It is offering role-based API access to a catalogue of services and automated workflows that support aggregating, generating, and sharing protected health information across communities of medical patients, practitioners and organizations. The company is facing the typical problem of software start-ups: The need of entering a competitive market with innovative product ideas with the concurrent goal of short term revenue generation.

Minimum viable products (MVPs) are one promising option to overcome the innovation risk challenge. The idea is concentrating on a smaller set of features offered in a product to a smaller set of customers. Duc and Abrahamsson [3] have emphasized the importance of MVPs in particular for start-up companies. While there is quite consensus on the potential benefit of MVP, not so much is known how to really achieve these products. In the context of the Brightsquid, development of a MVP is controlled by restricting resource allocation for each feature and highly dependent on interactive improvement. In this context, quick and accurate prediction of effort and the risk introduced by a change request is highly important.

- *M. Nayebi, S. J. Kabeer, and G. Ruhe are with the Software Engineering Decision Support Laboratory, University of Calgary, Canada E-mail: {mnayebi, shaikhjeeshan.kabeer, ruhe}@ucalgary.ca*
- *C. Carlson and F. Chew are with the BrightSquid company, Canada E-mail: {chris.carlson, francis}@brightsquid.com*

2 Challenge: Formal text similarity is not enough!

Brightsquid's software development life-cycle integrates Scrum methodology supported by ISO13485 based Quality Management System policies and procedures. Change requests management is of core importance in this highly adaptive process. For each sprint, the team decides which change requests are processed and in which order. Change requests are entered by the project manager in `Jira` platform and have a textual summary and description with average length of 6.25 and 23.7 words, respectively. For predicting the impact of change requests (identifying which files would impacted by a change request), our work relies on a similarity assumption:

`IF` two change requests CR_i and CR_j are textually similar `THEN` the files impacted by the change requests are similar `AND` higher similarity correlates with a higher number of impacted files being in common between CR_i and CR_j.

The assumption is adapted from analogy-based reasoning and is justified by observations made on files impacted by former change requests. The applicability of the hybrid labeling approach does not require a validation of its formal correctness. We explored and analyzed a variety of Natural Language Processing (NLP) techniques, similarity measures (Cosine, BM25 [6]), and benchmarked with different similarity thresholds. In Table 1, based on analysis of 169

TABLE 1
Accuracy of different models for predicting impact of change requests using textual similarity (*baseline*).

Technique	Precision	Recall	F1
BOW	0.35	0.3	0.32
BOW + Stopword	0.31	0.36	0.33
BOW + Stopword + Lemmatization	0.43	0.39	*0.41*
LDA	0.32	0.48	0.38
LDA + Stopword	0.33	0.53	0.40
LDA + Stopword + Lemmatization	0.29	0.49	0.36

change requests, we summarize the performance of some of these techniques using evaluation measures from [5]. As both precision (files predicted to be impacted are actually impacted) and recall (completeness of finding impacted files) are important, the achieved accuracy of predictions was insufficient from the company's perspective. In the best case, by combining bag of words (BOW) with customizing stop words and lemmatization, we achieved a F1-score (harmonic means between precision and recall) of 0.41 only.

The unsatisfactory results motivated us to discuss the semantic similarity of change requests with developers and project managers. Discussing numerous change requests in brainstorming sessions with BrightSquid, we observed that the developers use some keywords (e.g., code functions or service layers) to describe the similarity between change requests. Several of these keywords does not exist in the summary of change requests, and others are filtered out by frequency based techniques (such tf-idf) and probabilistic models (such as LDA [1]) as they were used in a few specific change requests.

3 Analytical Resolution and Effectiveness

Labels in social coding are a form of metadata that help people finding a theme between different code artifacts such as commit messages and change requests. Labelling is a common technique for social developers and social media users because they use tags (also known as hashtags). Previous studies showed that code item tagging could be easily adopted by development teams [8]. In development platforms with social aspects such as `GitHub` and `Jira`, labeling is possible and promoted as a way for categorizing change requests and informally assigning them to a component or release.

We performed a small experiment with the aim to understand the representativeness and limitations of analyzing similarity just based on textual description. We asked three developers to manually label ten randomly selected change requests (manual labeling). We also applied topic modeling (LDA) using seven words to describe each topic. Comparing manual labels with machine labels, we found that 51.3% of human and machine labels are the same. We noticed that, developers often add labels that do not exist in the textual description of the change request. Such labels often reflect the architecture layer, user layers, and even code methods. For example:

> ***Change request:*** *SQL Migration for lab request, comments and docs*
> ***Machine labels:*** *SQL, Migrate, Lab, request, comment, doc*
> ***Manual labels:*** *SQL, Migrate, StandardRX, LabRequest, PatientCaseComment*

The difference between human labels and machine generated labels provoked us to study hybrid labeling (using both labels). The analysis of these labels was done for 169 change requests within one project and over 12 months. The performance of these prediction models is shown in Table 2. Comparing the results, a prediction model based on developers' labels perform 7% better in their F1-score compared to machine labeling. Combining machine and developer's labels improved the F1-score by 13%.

TABLE 2
Accuracy of prediction model for scoping file changes based on labelling.

Technique	Precision	Recall	F1
Machine labels (LDA)	0.33	0.54	0.41
Manual labels	0.41	0.58	0.48
Hybrid (machine + manual) labels	0.54	0.70	0.61

4 What comes next?

Our study showed that expert's knowledge added to machine labeling helps in improving the prediction of change impact analysis. Labeling proved to be useful for reminding and re-finding [7], organizing and better performing better collaboration [8]. For Brightsquid and from supply side, the results encapsulate the value they are looking for in the MVP development process which is *helping them to more quickly and accurately predict the effort and the risk introduced by a change request*.

Based on these results, we designed a labeling recommendation system to suggest up to 10 labels for each change request using LDA. Each developer can select most relevant labels or add other labels based on personal expertise. There are multiple opportunities for improvement. First, while we relied on the study by Diaz et al. [2] for recommending tags using LDA, this might not be the best technique for extracting automatic labeling [9]. Second, as different software artifacts such as code and change requests often have a hierarchical structure (e.g., stories, sub-stories, and sub-technical tasks), exploiting inheritance of labels may add more context for similarity analysis. Third, while this study was limited to the performance of change impact prediction, a more detailed study would focus on using labels for analogy based reasoning for other domains of software project and product planning and prediction [4].

References

[1] D. M. Blei, A. Y. Ng, and M. I. Jordan. Latent dirichlet allocation. *Journal of machine Learning research*, 3(Jan):993–1022, 2003.

[2] E. Diaz-Aviles, M. Georgescu, A. Stewart, and W. Nejdl. Lda for on-the-fly auto tagging. In *Proc. Conference on Recommender Systems*, pages 309–312. ACM, 2010.

[3] A. Duc and P. Abrahamsson. Minimum viable product or multiple facet product? the role of mvp in software startups. *LNBIP*, 251:118–130, 2016. cited By 0.

[4] M. Nayebi, G. Ruhe, R. C. Mota, and M. Mufti. Analytics for software project management–where are we and where do we go? In *ASE Workshop Action15*, pages 18–21. IEEE, 2015.

[5] D. M. Powers. Evaluation: from precision, recall and f-measure to roc, informedness, markedness and correlation. 2011.

[6] Z. Shi, J. Keung, and Q. Song. An empirical study of bm25 and bm25f based feature location techniques. In *Proc. Workshop on Innovative Software Development Methodologies and Practices*, pages 106–114. ACM, 2014.

[7] M.-A. Storey, J. Ryall, J. Singer, D. Myers, L.-T. Cheng, and M. Muller. How software developers use tagging to support reminding and refinding. *IEEE TSE*, 35(4):470–483, 2009.

[8] C. Treude and M.-A. Storey. Work item tagging: Communicating concerns in collaborative software development. *IEEE TSE*, 38(1):19–34, 2012.

[9] X. Xia, D. Lo, X. Wang, and B. Zhou. Tag recommendation in software information sites. In *Proc. Conference MSR*, pages 287–296. IEEE, 2013.