Automated Recovery of Issue-Commit Links Leveraging Both Textual and Non-textual Data

Pooya Rostami Mazrae
Software Engineering Lab, University of Mons, Belgium
pooya.rostamimazrae@umons.ac.be

Issues and commits are commonly used during collaborative distributed development of (mainly open source) projects, using social coding platforms such as GitHub or GitLab, and issue and bug tracking systems such as Jira and Bugzilla. Issue reports mainly focus on requests from end-users or project contributors for new features, changes in existing functionality, bug fixes, performance improvements, and so on. Commits contain source code changes to address these reported issues.

Contemporary social coding platforms such as GitHub aim to provide integrated support for issues, commits, pull requests and other kinds of coding activities. For a variety of reasons, many companies and open source project communities (Mozilla, Apache and OpenStack) continue to use separate issue trackers, bug trackers and code reviewing systems, even if their code is hosted on a social coding platform like GitHub. Issues and bugs are often tracked in other systems such as Bugzilla, Jira and Storyboard.

When a developer commits a code change, it is good practice to explicitly mention in the commit message which issue or bug the commit relates to. Unfortunately, this is seldomly the case due to deadline pressure, lack of motivation, etc. Indeed, Ruan et al. [1] quantified the prevalence of missing issue-commit links based on an analysis of over half a million GitHub issues, finding that only 42.2% of issues were linked to corresponding commits.

Recovering issue-commit links is important for improving bug prediction solutions, bug assignment, feature location techniques, and other software maintenance tasks. It is also useful to evaluate maintenance effort and software quality. Thus, an automated method for recovering links between commits and their corresponding issues can be of high value.

In this work, we introduce *Hybrid-Linker*[1], a novel approach to address the aforementioned problem. This work has been published and presented at ICSME 2021 [2].

We first identify all relevant information from issues and commits and then perform feature engineering to extract the most important ones. Hybrid-Linker exploits both textual and non-textual data from the issues and commits to achieve higher performance than DeepLink [1] and FRLink [3]. Textual information includes the issue title, description, commit messages and code difference. Non-textual information includes various characteristics such as the issue author, issue type (bug, feature, task), issue status (open, closed, or resolved), the committer and commit time. Incorporating non-textual data enables Hybrid-Linker to exploit this knowledge when little textual information is available (e.g., there are no commit messages), or when there are few similarities between the description of an issue and the commit message.

[1]https://github.com/MalihehIzadi/hybrid-linker

TABLE I: execution time for each project on our hardware

Project	Hybrid-Linker	DeepLink	Project	Hybrid-Linker	DeepLink
Beam	35m	19h	Flink	2h	3d
Freemarker	11.5s	30m	Airflow	25m	7h
Arrow	35m	6h	Netbeans	7m	25d
Ignite	22m	13.5d	Isis	28m	23h
Groovy	54m	13h	Cassandra	33m	6h
Ambari	4h	7.5d	Calcite	31m	6h

We train a hybrid model consisting of two classifier components (i.e. non-textual and textual classifiers) and a module to achieve the best linear composition of these classifiers. The non-textual component is an ensemble of a Gradient Boosting model and a XGBoost model. The textual component is created using TF-IDF word embeddings and a single Gradient Boosting model.

We evaluated Hybrid-Linker against two baseline methods, FRLink [3] and DeepLink [1], for 12 Apache projects with different characteristics. These projects were sampled from bigger dataset gathered in 20-MAD [4]. Our evaluation results on these 12 projects show that Hybrid-Linker outperforms FR-Link [3] and DeepLink [1] in terms of F1-measure by 31.3%, and 41.3% respectively. Moreover, the proposed approach shows extensive improvements in terms of required training time in comparison with the more recent DeepLink method [1] when using systems with less computational resources. All our experiments, were performed used a machine with 32GB memory and a 4-core Intel i7-7700k 4.2G processor. Table I compares the training time for the selected projects.

References

[1] H. Ruan, B. Chen, X. Peng, and W. Zhao, "Deeplink: Recovering issue-commit links based on deep learning," *Journal of Systems and Software*, vol. 158, p. 110406, 2019.

[2] P. R. Mazrae, M. Izadi, and A. Heydarnoori, "Automated recovery of issue-commit links leveraging both textual and non-textual data," in *2020 IEEE International Conference on Software Maintenance and Evolution (ICSME)*. IEEE, 2021.

[3] Y. Sun, Q. Wang, and Y. Yang, "Frlink: Improving the recovery of missing issue-commit links by revisiting file relevance," *Information and Software Technology*, vol. 84, pp. 33–47, 2017.

[4] M. Claes and M. V. Mäntylä, "20-mad: 20 years of issues and commits of mozilla and apache development," in *Proceedings of the 17th International Conference on Mining Software Repositories*, 2020, pp. 503–507.