Semantically-enriched Jira Issue Tracking Data

Themistoklis Diamantopoulos, Dimitrios-Nikitas Nastos, and Andreas Symeonidis
Electrical and Computer Engineering Dept., Aristotle University of Thessaloniki
Thessaloniki, Greece
thdiaman@ece.auth.gr, diminast@ece.auth.gr, symeonid@ece.auth.gr

***Abstract*—Current state of practice dictates that software developers host their projects online and employ project management systems to monitor the development of product features, keep track of bugs, and prioritize task assignments. The data stored in these systems, if their semantics are extracted effectively, can be used to answer several interesting questions, such as finding who is the most suitable developer for a task, what the priority of a task should be, or even what is the actual workload of the software team. To support researchers and practitioners that work towards these directions, we have built a system that crawls data from the Jira management system, performs topic modeling on the data to extract useful semantics and stores them in a practical database schema. We have used our system to retrieve and analyze 656 projects of the Apache Software Foundation, comprising data from more than a million Jira issues.**

***Index Terms*—Mining Software Repositories, Task Management, Jira Issues, Topic Modeling, BERT**

I. Introduction

The introduction of the open-source paradigm and the evolution of online services has lately transformed software development into a highly collaborative process. More and more teams nowadays use online facilities both to host their code and software artefacts, as well as to effectively monitor the software development process. Indicatively, GitHub, at the time of writing, hosts more than 200 million repositories from more than 65 million developers[1]. Effective collaboration, however, is not only limited to developing source code artefacts, but is also required over multiple axes, including (but not limited to) documentation writing, feature design and discussion, bug resolution, etc.

As a result, several issue tracking systems and project management services are deployed online to allow effective monitoring of the software development process. This collaborative paradigm is quite popular in contemporary software teams, and is also especially preferred by large open-source organizations, such as the Eclipse Foundation and the Mozilla Foundation (that both use Bugzilla[2]), or the Apache Software Foundation (that uses Jira[3]). When having multiple contributors and multiple projects, these project management services provide effective ways to keep track of tasks, prioritize and assign new features, resolve bugs, and set deadlines for crafting new releases.

Apart from monitoring and auditing the progress of software projects, the data recorded in these systems can also be harnessed to confront multiple challenges. For instance, extracting and mining this type of data can be useful to automatically determine the importance and/or priority of new tasks [1]–[3] or the severity of bugs [4]–[8], to recommend the most suitable developer for fixing a newly found bug [9]–[14], to extract the roles and behavior of different contributors [15]–[19], or even to quantify the software development process and investigate how the productivity of a team is influenced by the current workload [20]–[23]. Lately, these challenges are widely confronted using semantics-enabled methods, such as topic modeling [24]–[27], word embeddings [13], [28], or even more complex models based on knowledge-extraction network architectures [29].

Concerning the data sources employed by the aforementioned approaches, several of them employ a limited number of projects extracted from individual Bugzilla/Jira installations [1]–[5], [8], [10]–[12]. And although there are certain datasets that extract issues from multiple projects [30]–[32], they are not always practical and/or up to date (e.g. the Jira Repository Dataset [30] is tailored to sentiment analysis and is last updated in 2016), while they are often focused on the textual information of issues [32] or even their connection with commits [31], without incorporating semantics.

In this context, we have built a system that crawls the Jira infrastructure of the Apache Software Foundation[4], analyzes the data of all projects and stores them in a database schema suitable for answering multiple questions relevant to the software development process. Our analysis further includes the extraction of topics using the BERTopic topic modeling technique [33], to semantically enrich the issue data. The dataset comprises more than a million issues from approximately 650 projects, while our issue retrieval tool supports incremental updates, ensuring that the data are always up-to-date. Moreover, the dataset that we extracted is available as a MongoDB dump, thus allowing easy set-up and advanced querying capabilities.

II. Architecture and Tools

Fig. 1 depicts the architecture of our platform, which has two modules, the *Jira Apache Downloader* and the *Jira Topic Extractor*. These tools, which are analyzed in the following paragraphs, are available online[5] to allow full reproducibility.

[1]https://github.com/
[2]https://www.bugzilla.org/
[3]https://www.atlassian.com/software/jira
[4]https://issues.apache.org/jira
[5]The web addresses of the Jira Apache Downloader and the Jira Topic Extractor are https://github.com/AuthEceSoftEng/jira-apache-downloader and https://github.com/AuthEceSoftEng/jira-topic-extractor, respectively.

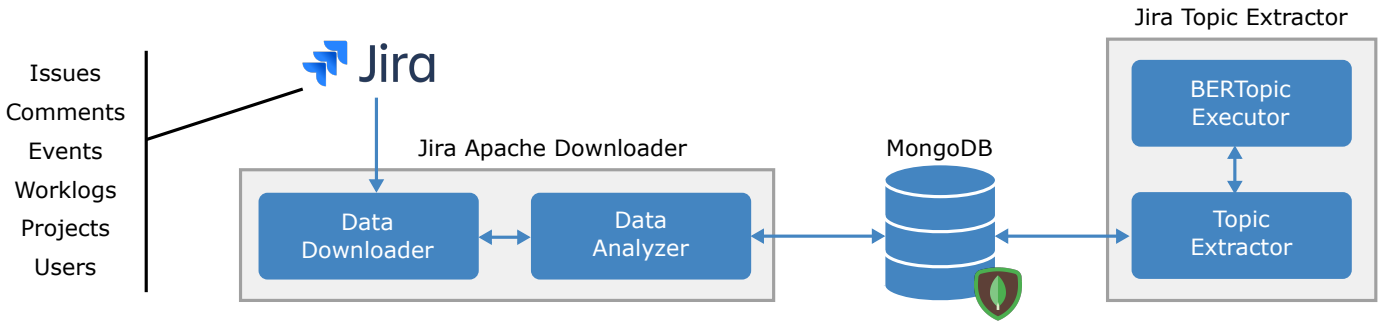

Fig. 1. Architecture overview of the system.

A. Jira Apache Downloader

The Jira Apache Downloader comprises two interoperating components, the Data Downloader and the Data Analyzer. The *Data Downloader* uses a token as credentials for the Jira installation and calls the Jira server API in order to retrieve the issues of a given project. Version 2 of the Jira API is what we may call *issue-centric*, meaning that issue objects practically contain all available information, i.e. contributors, projects, comments, etc. are all included in the issues. The advantage of this solution is that documents are self-contained, however there is a high degree of redundancy and the information is not efficiently indexed.

As a result, we developed the *Data Analyzer* component, which processes the retrieved issues and builds one collection in a Mongo database for each of the Jira objects shown at the left of Fig. 1. In specific, for each issue, we extract the users, the events, the comments, and the worklogs, we populate the corresponding collections and create links between the documents. For instance, for any user found in issues, events or worklogs (i.e. reporter or assignee in issues, author in events, etc.), we create a user document in the database, and connect it to the issue (or event or comment or worklog) via the user id. Furthermore, the id of the originating issue and the name of the project are stored and indexed in all documents, to allow fast retrieval of the data of a project or of an issue.

Before posting a new document, we also check whether it already exists (and if it has to be updated) and also process the document fields in order to make sure that they are correctly formatted (e.g. converting string dates to date objects). When receiving a request to download the issues of a project, the database is first queried to determine whether the project already exists. If the project already exists, then we update only the documents that have been changed since the last time that we crawled the project.

B. Jira Topic Extractor

Upon retrieving the Jira issues and storing them in Mongo, the next step is to extract useful semantics from them. To do so, we have used BERTopic [33], a topic modeling technique that employs transformers and class-based TF-IDF to extract semantic topics from a corpus of documents. One of the basic advantages of this topic modeling method against conventional models like LDA [34] is the usage of powerful BERT-based [35] language models. These models generate contextual representations for the documents and enable BERTopic to identify semantic relationships between them, while conventional ones are based on bag-of-words representations, which are inherently less effective for extracting semantics. BERTopic is executed for each project in 3 steps. First, for every issue of the project, we extract its title (i.e. the summary field) and description, concatenate them and give them as input to BERTopic, which uses a BERT model to create an embedding representation for each issue. The next step is to reduce the dimensionality of these embeddings and create clusters of semantically similar documents. And, finally, using c-TF-IDF, a class-based TF-IDF procedure, BERTopic calculates the importance of specific terms in the clusters and extracts topic representations.

We use the issues of 497 projects that have an adequate number of issues to implement BERTopic (which, in most cases, required at least 100 issues) and extract different topics from them. The BERT model we use for the embeddings generation is SBERT [36], which is the default option chosen by BERTopic. Furthermore, HDBSCAN [37] is used for clustering, while UMAP [38] is used for dimensionality reduction. The number of topics extracted is selected automatically by BERTopic, while at the end of execution any topics with high similarity according to their c-TF-IDF representations are merged by using HDBSCAN again. The outcome of applying BERTopic to the issues of each project includes the topic representations, the top terms per topic, and the distribution between topics and issues, i.e. the probabilities for every issue of the project to belong to each extracted topic.

III. Dataset Construction and Usage

The schema designed for the semantically-enriched Jira issue tracking data is shown in Figure 2. The *issues* collection involves all elements stored by Jira for a specific issue, including e.g. the issue title, its description, its creation/update dates, etc. Furthermore, it includes a field “projectname”, which connects this issue to the corresponding project (collection *projects*) similarly to a foreign key in relational database terms. Note also that both this and all other foreign-key like connections are indexed, allowing for fast queries over the

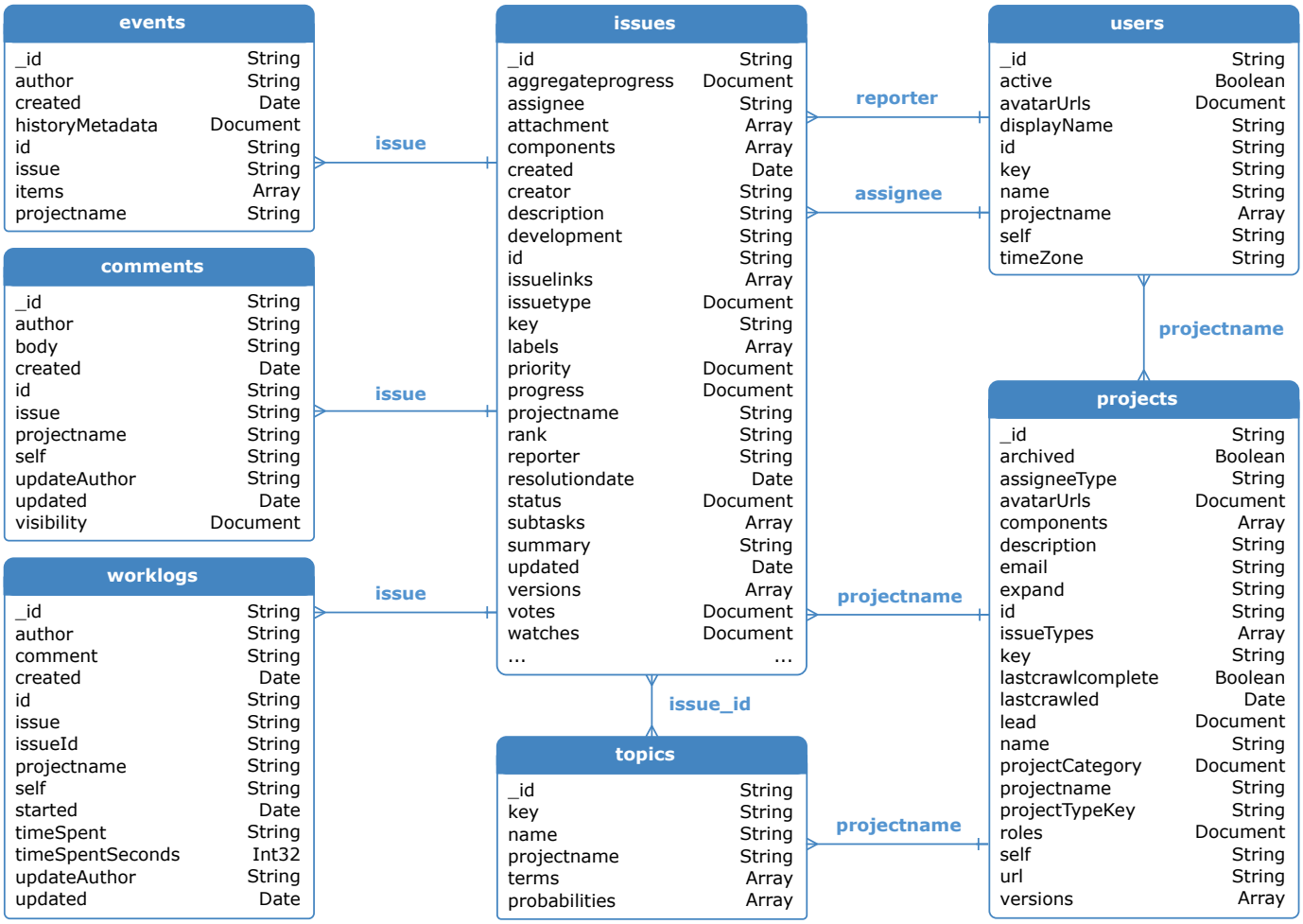

Fig. 2. Schema of the MongoDB used to store Jira projects (not all connections are shown).

data. The reporter and the assignee of the issue are connected to the *users* collection, therefore reducing the redundancy of adding all their fields (e.g. name, avatar) to each issue. Users are also connected to projects with a many-to-many relation.

Concerning the *events*, *comments*, and *worklogs* collections, all three are connected to a specific issue as well as to a specific project. The events involve the changes performed in an issue; apart from the dates and the people involved, each event stores an array of "items", i.e. modifications performed in the issue (e.g. a summary update or a status change). The comments also have created/updated dates and authors. Finally, the worklogs allow Jira contributors to keep track of how much work they have done for a specific issue. So, they include the relevant fields for the ids, the dates, the authors, and the time spent for the corresponding action.

The *topics* collection involves all the topics extracted from the issues of each project and the relevant information about them. Specifically, for each topic we store its name, its top terms as well as the 'projectname' field which refers to the project this topic belongs to. Every topic also has an array of probabilities which stores the probability of each issue to belong to this topic (i.e. each element contains the id of an issue and the corresponding probability). Probabilities lower than a 10^{-10} threshold are dropped for performance reasons. Thus, one can easily get all the topics related to a specific project and gain insight as to the challenges confronted by it.

Fig. 3 depicts the topics extracted from the issues of the FTPSERVER project. For each topic we may see the 2D embedding representations of the assigned issues, generated with dimensionality reduction using UMAP, in different colors. The topics seem to be well defined; for instance, topic 1 is mostly related to server connections, which is a rather expected function of an ftp server. Similarly, we may identify topics that are relevant to the Maven build system (topic 2), the Spring Framework (topic 6), etc. Interestingly, we see that certain topics are more cohesive, as they may relate to a very specific component (e.g. topic 7, which concerns data I/O and sockets), while others may be more generic as they are used throughout the project (e.g. topic 0, which relates to file management).

Finally, our dataset is available online in the form of a MongoDB data dump and a folder with our trained models[6]. Certain statistics are shown in Table I.

[6]https://doi.org/10.5281/zenodo.5665895

Fig. 3. Example of topics extracted by the FTPSERVER project, where each topic is represented with a different color.

TABLE I
DATASET STATISTICS.

Metric	Value
Number of Projects	656
Number of Users	147610
Number of Issues	1013964
Number of Comments	4639882
Number of Events	7503864
Number of Worklogs	414893
Number of Topics	10510
Data Size	20.64GB (5.67GB DB-compressed)

IV. Impact and Research Directions

Our dataset can be used to confront several challenges in current research. For instance, it can be employed for the problem of automated bug triaging. Determining the most suitable developer for undertaking a task or fixing a newly found bug has been extensively researched, and is typically considered as a challenge that can benefit from the incorporation of semantics [13], [28], [29], and even specifically from the extraction of topics [24]–[27]. For instance, contemporary methods extract topics from issues, and when a new issue arrives, they first determine its topic to find similar issues, and subsequently use their features to determine the most suitable developer for fixing it. Other similar challenges that can be addressed include predicting the priority of issues [1], [2] or the severity of bugs [4]–[8]. Furthermore, our dataset is not necessarily limited to bugs, as it comprises issues that may describe features, requirements, or even documentation. As a result, it could be used in a broader context, e.g. for feature assignment [14] or task importance prediction [3].

Another interesting research direction is that of automated role (or behavior) extraction [17]–[19]. Approaches in this field often apply semantic analysis on projects to determine the areas of projects and/or components that are associated with different developers [15]. Since the issues of our dataset are also assigned to components, such an analysis would be possible. Furthermore, given the information from the extracted topics, one could create personalized developer profiles that would describe the area of expertise for each developer [16] in comprehensible terms. Finally, since our dataset includes also events and worklogs from different projects, it could even be used for quantifying the productivity of software development teams with respect to the current workload [20], [21].

V. Conclusions

Mining project management data from issue tracking systems can offer useful insights for researchers and practitioners in software engineering. In this work, we have worked towards this direction by presenting a system and a dataset of issues extracted from the Jira tracker of Apache. Furthermore, we have used topic modeling to extract semantic information that can be used along with the contribution information of issues in order to address multiple research challenges. As future work, we will focus on comparing BERTopic with other methods that extract semantics from issues as well as on demonstrating how these semantics can increase the effectiveness of bug triaging and bug priority/severity methods. Finally, we plan to augment our dataset by including also source code/commit information, drawn by the GitBox and GitHub repository services of the Apache Software Foundation, to further enhance traceability and support interesting research directions.

REFERENCES

[1] Y. Tian, D. Lo, X. Xia, and C. Sun, "Automated Prediction of Bug Report Priority Using Multi-Factor Analysis," *Empirical Softw. Engg.*, vol. 20, no. 5, pp. 1354–1383, Oct 2015.

[2] J. Kanwal and O. Maqbool, "Bug Prioritization to Facilitate Bug Report Triage," *Journal of Computer Science and Technology*, vol. 27, no. 2, pp. 397–412, Mar 2012.

[3] T. Diamantopoulos, C. Galegalidou, and A. L. Symeonidis, "Software task importance prediction based on project management data," in *16th International Conference on Software Technologies*, ser. ICSOFT 2021. Held Online: SciTePress, 07 2021, pp. 269–276.

[4] A. Lamkanfi, S. Demeyer, E. Giger, and B. Goethals, "Predicting the Severity of a Reported Bug," in *2010 7th IEEE Working Conference on Mining Software Repositories*. IEEE Press, 2010, pp. 1–10.

[5] N. K. S. Roy and B. Rossi, "Towards an Improvement of Bug Severity Classification," in *Proceedings of the 2014 40th EUROMICRO Conference on Software Engineering and Advanced Applications*, ser. SEAA '14. USA: IEEE Computer Society, 2014, pp. 269–276.

[6] T. Menzies and A. Marcus, "Automated Severity Assessment of Software Defect Reports," in *2008 IEEE International Conference on Software Maintenance*, ser. ICSM 2008. IEEE Press, 2008, pp. 346–355.

[7] Y. Tian, D. Lo, and C. Sun, "Information Retrieval Based Nearest Neighbor Classification for Fine-Grained Bug Severity Prediction," in *Proceedings of the 2012 19th Working Conference on Reverse Engineering*. USA: IEEE Computer Society, 2012, pp. 215–224.

[8] G. Yang, T. Zhang, and B. Lee, "Towards Semi-Automatic Bug Triage and Severity Prediction Based on Topic Model and Multi-Feature of Bug Reports," in *Proceedings of the 2014 IEEE 38th Annual Computer Software and Applications Conference*, ser. COMPSAC '14. USA: IEEE Computer Society, 2014, pp. 97–106.

[9] V. Matsoukas, T. Diamantopoulos, M. Papamichail, and A. Symeonidis, "Towards analyzing contributions from software repositories to optimize issue assignment," in *2020 IEEE International Conference on Software Quality, Reliability and Security*, ser. ICSOFT 2020. Vilnius, Lithuania: IEEE, 07 2020, pp. 243–253.

[10] D. Ĉubranic and G. C. Murphy, "Automatic bug triage using text categorization," in *Proceedings of the Sixteenth International Conference on Software Engineering & Knowledge Engineering*, ser. SEKE'04, Citeseer. DBLP, 2004, pp. 92–97.

[11] J. Anvik, L. Hiew, and G. C. Murphy, "Who Should Fix This Bug?" in *Proceedings of the 28th International Conference on Software Engineering (ICSE '06)*. New York, NY, USA: ACM, 2006, pp. 361–370.

[12] X. Xia, D. Lo, X. Wang, and B. Zhou, "Accurate developer recommendation for bug resolution," in *Proceedings of the 2013 20th Working Conference on Reverse Engineering*, ser. WCRE 2013. IEEE, Oct 2013, pp. 72–81.

[13] S.-R. Lee, M.-J. Heo, C.-G. Lee, M. Kim, and G. Jeong, "Applying deep learning based automatic bug triager to industrial projects," in *Proceedings of the 2017 11th Joint Meeting on Foundations of Software Engineering*, ser. ESEC/FSE 2017. New York, NY, USA: Association for Computing Machinery, 2017, p. 926–931.

[14] K. Christidis, F. Paraskevopoulos, D. Panagiotou, and G. Mentzas, "Combining Activity Metrics and Contribution Topics for Software Recommendations," in *Proceedings of the Third International Workshop on Recommendation Systems for Software Engineering*, ser. RSSE '12. Piscataway, NJ, USA: IEEE Press, 2012, pp. 43–46.

[15] M. D. Papamichail, T. Diamantopoulos, V. Matsoukas, C. Athanasiadis, and A. L. Symeonidis, "Towards extracting the role and behavior of contributors in open-source projects," in *14th International Conference on Software Technologies (ICSOFT)*, ser. ICSOFT 2019. Prague, Czech Republic: SciTePress, 07 2019, pp. 536–543.

[16] G. J. Greene and B. Fischer, "CVExplorer: Identifying Candidate Developers by Mining and Exploring Their Open Source Contributions," in *Proceedings of the 31st International Conference on Automated Software Engineering (ASE)*. NY, USA: ACM, 2016, pp. 804–809.

[17] S. Li, H. Tsukiji, and K. Takano, "Analysis of Software Developer Activity on a Distributed Version Control System," in *Proceedings of the 30th International Conference on Advanced Information Networking and Applications Workshops*. IEEE, 2016, pp. 701–707.

[18] S. Onoue, H. Hata, and K.-i. Matsumoto, "A Study of the Characteristics of Developers' Activities in GitHub," in *Proceedings of the 20th Asia-Pacific Software Engineering Conference*. USA: IEEE, 2013, pp. 7–12.

[19] A. Sarma, X. Chen, S. Kuttal, L. Dabbish, and Z. Wang, "Hiring in the Global Stage: Profiles of Online Contributions," in *Proceedings of the 11th IEEE International Conference on Global Software Engineering*, ser. ICGSE 2016. IEEE, Aug 2016, pp. 1–10.

[20] G. Gousios, E. Kalliamvakou, and D. Spinellis, "Measuring developer contribution from software repository data," in *Proceedings of the 2008 International Working Conference on Mining Software Repositories*, ser. MSR '08. NY, USA: ACM, 2008, pp. 129–132.

[21] J. Lima, C. Treude, F. F. Filho, and U. Kulesza, "Assessing developer contribution with repository mining-based metrics," in *Proceedings of the 2015 IEEE International Conference on Software Maintenance and Evolution (ICSME)*. USA: IEEE, 2015, pp. 536–540.

[22] T. Diamantopoulos, M. Papamichail, T. Karanikiotis, K. Chatzidimitriou, and A. Symeonidis, "Employing contribution and quality metrics for quantifying the software development process," in *IEEE/ACM 17th International Conference on Mining Software Repositories*, ser. MSR '20. Seoul, South Korea: IEEE, 06 2020, p. 558–562.

[23] K. D. Maxwell and P. Forselius, "Benchmarking software-development productivity," *IEEE Softw.*, vol. 17, no. 1, pp. 80–88, Jan 2000.

[24] S. N. Ahsan, J. Ferzund, and F. Wotawa, "Automatic software bug triage system (bts) based on latent semantic indexing and support vector machine," in *Proceedings of the 2009 Fourth International Conference on Software Engineering Advances*, ser. ICSEA '09. USA: IEEE Computer Society, 2009, p. 216–221.

[25] G. Yang, T. Zhang, and B. Lee, "Towards semi-automatic bug triage and severity prediction based on topic model and multi-feature of bug reports," in *Proceedings of the 2014 IEEE 38th Annual Computer Software and Applications Conference*, ser. COMPSAC '14. USA: IEEE Computer Society, 2014, p. 97–106.

[26] H. Naguib, N. Narayan, B. Brügge, and D. Helal, "Bug report assignee recommendation using activity profiles," in *Proceedings of the 10th Working Conference on Mining Software Repositories*, ser. MSR '13. IEEE Press, 2013, p. 22–30.

[27] X. Xia, D. Lo, Y. Ding, J. M. Al-Kofahi, T. N. Nguyen, and X. Wang, "Improving automated bug triaging with specialized topic model," *IEEE Trans. Softw. Eng.*, vol. 43, no. 3, p. 272–297, 2017.

[28] S. Mani, A. Sankaran, and R. Aralikatte, "Deeptriage: Exploring the effectiveness of deep learning for bug triaging," in *Proceedings of the ACM India Joint International Conference on Data Science and Management of Data*, ser. CoDS-COMAD '19. New York, NY, USA: Association for Computing Machinery, 2019, p. 171–179.

[29] H. Mohsin, C. Shi, S. Hao, and H. Jiang, "Span: A self-paced association augmentation and node embedding-based model for software bug classification and assignment," *Know.-Based Syst.*, vol. 236, no. C, 2022.

[30] M. Ortu, G. Destefanis, B. Adams, A. Murgia, M. Marchesi, and R. Tonelli, "The jira repository dataset: Understanding social aspects of software development," in *Proceedings of the 11th International Conference on Predictive Models and Data Analytics in Software Engineering*, ser. PROMISE '15. New York, NY, USA: Association for Computing Machinery, 2015.

[31] F. Trautsch, S. Herbold, P. Makedonski, and J. Grabowski, "Addressing problems with replicability and validity of repository mining studies through a smart data platform," *Empirical Softw. Engg.*, vol. 23, no. 2, p. 1036–1083, 2018.

[32] R. Vieira, A. da Silva, L. Rocha, and J. a. P. Gomes, "From reports to bug-fix commits: A 10 years dataset of bug-fixing activity from 55 apache's open source projects," in *Proceedings of the Fifteenth International Conference on Predictive Models and Data Analytics in Software Engineering*. New York, NY, USA: ACM, 2019, p. 80–89.

[33] M. Grootendorst, "Bertopic: Neural topic modeling with a class-based tf-idf procedure," *arXiv preprint arXiv:2203.05794*, 2022.

[34] D. M. Blei, A. Y. Ng, and M. I. Jordan, "Latent Dirichlet Allocation," *J. Mach. Learn. Res.*, vol. 3, p. 993–1022, 2003.

[35] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding," *CoRR*, vol. abs/1810.04805, 2018.

[36] N. Reimers and I. Gurevych, "Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks," *CoRR*, vol. abs/1908.10084, 2019.

[37] L. McInnes, J. Healy, and S. Astels, "hdbscan: Hierarchical density based clustering," *The Journal of Open Source Software*, vol. 2, no. 11, p. 205, Mar. 2017.

[38] L. McInnes, J. Healy, and J. Melville, "UMAP: Uniform Manifold Approximation and Projection for Dimension Reduction," *arXiv preprint arXiv:1802.03426*, 2018.