Exploring Popular Software Repositories: A Study on Sentiment Analysis and Commit Clustering

Bianca Puerta Rocha Vieira[a] and Rogério Eduardo Garcia[b]
Department of Mathematical and Computer Science, São Paulo State University (UNESP), R. Roberto Símonsen, 305, Centro Educacional, Pres. Prudente, SP, Brazil

Keywords: Sentiment Analyses, Mining Software Repositories, Knowledge Discovery.

Abstract: The software repositories store data and metadata about the project development, including commits, which record user modifications to projects and their metadata, such as the user responsible for the commit, date, time, and others. The programmer can register a comment to inform the modification content, its purpose, requester, motivation, and useful data. Focusing on those comments, this paper proposes using comments to group the commits and construct a sentiment analysis regarding the messages. The main purpose is to analyze those messages, both by the groups and the sentiments expressed, to understand them (what sort of sentiment they express). Opinions are central to almost all human activities and are key influences on our behaviors. Beliefs, perceptions of reality, and choices made are conditioned upon sentiments. Therefore, understanding how the developers, especially programmers, feel about a task might be useful in analyzing progress and interaction among people and artifacts (source code). In this paper, we present initial analyses of data and metadata from the twenty most popular software repositories, written in five popular programming languages. We stated five research questions and answered them, pointing out further investigations.

1 INTRODUCTION

The software repositories are valuable information sources about the project's development. The data kept by repositories might be mined to supply information to the project's managers. Still, they are also used to answer important research questions about several perspectives (Zafar et al., 2019) such as dealing with technical debt (de Lima et al., 2022), recommendation (Nguyen et al., 2023), and understanding the time to first response in GitHub pull request (Hasan et al., 2023) or how software developers use repository resources (Kinsman et al., 2021).

In software repositories based on Git version control, the commits store the modifications made by a user at a particular moment, in a specific code spot, and with a comment about the change made. This comment characterizes a message for future analysis – in this paper, the term **message** refers to these comments. There is a convention of commit types[1] to assist the construction of automated tools that process data collected in the repository. However, not all users follow the convention when registering messages related to commits in a repository. Thus, Cosentino et al. (2017) argue that it is possible to find repositories with all messages according to the convention, mixed or not. They also assert that GitHub is the best platform that allows access to a repository of data and its artifacts, as in addition to being one of the most popular platforms, it provides an interface (API) to get the data publicly. It has been widely used in studies and metadata analyses in project management and software development research.

On the other hand, more and more analytical methods have been applied to analyze data repositories. Data mining has been applied, generating a research area: Mining Software Repositories (MSR). MSR can be used to understand the team dynamics, the developer's behavior, and other purposes.

The repositories keep projects of several programming languages, and the community evaluates them according to the interest aroused in them (the project number of stars). It also can be studied, in addition to the relationships between commit data grouped by repository popularity and language used in the project. Five of the most popular languages and twenty of the most popular repositories from each language (the best evaluated by the community) were chosen for this study.

This paper aims to group commits, compare the sentiments obtained from commit metadata inside the

[a] https://orcid.org/0000-0001-6006-6093
[b] https://orcid.org/0000-0003-1248-528X
[1] https://www.conventionalcommits.org/en/v1.0.0/

Vieira, B. and Garcia, R.
Exploring Popular Software Repositories: A Study on Sentiment Analysis and Commit Clustering.
DOI: 10.5220/0012633400003690
Paper published under CC license (CC BY-NC-ND 4.0)
In *Proceedings of the 26th International Conference on Enterprise Information Systems (ICEIS 2024) - Volume 1*, pages 297-304
ISBN: 978-989-758-692-7; ISSN: 2184-4992
Proceedings Copyright © 2024 by SCITEPRESS – Science and Technology Publications, Lda.

groups generated, and compare with more data about the commit and the repository it belongs to. Specific goals: (1) Study the number of commits from each group that belongs to the programming language studied, comparing the proportion of groups inside the language. **RQ1: Do programming languages influence the distribution and group proportions?** (2) Study the relationship between the generated groups and the sentiments in the commit messages, analyzing the percent of each Sentiment inside the groups, as the groups represent a developing activity. **RQ2: Does the activity type influence the sentiments?** (3) Analyze the sentiments expressed in the commits comments inside the grouping by popularity (related to the stars of each project). **RQ3: Do the sentiments in the commits reflect the popularity of the project to which it belongs?** (4) Study the relationship between the percentage of each Sentiment in the commit messages and the programming languages used. **RQ4: What are the language's influences on the sentiments?** (5) Relate the repository's popularity with the generated groups. **RQ5: Do the activities (groups) relate to repositories popularity?**

In general, to achieve the specific goals, the following activities were performed: data extraction, data cleaning and normalization, clustering model development for grouping commit messages, sentiment analysis in commit groups, and related to the languages used and presentation of helpful information to establish the patterns that were found.

This paper is organized as follows: the related works are presented in Section 2. Next, in Section 3, the theoretical reference used to develop this project is presented. The methodology is presented in Section 4, describing the step-by-step activities developed. The analysis of results obtained is presented in Section 5. Finally, final considerations related to the objectives achieved are given in Section 6.

All queries, datasets and code used in this paper are available in the author's GitHub repository[2].

2 RELATED WORK

Ji et al. (2018) presents an analysis of bug-fix commits related to bug reports and secondary bug-fix commits that complement or adjust primary commits (primary because they were the first commits trying to solve the bug). They point out that secondary commits are often neglected, although their analysis can be beneficial. To classify complementary commits, a neural network is trained with commits related to pulling requests containing keywords related to bug-fix. After detecting the bug-fix commits, the relevance between commits is analyzed. The relevance shows if they share the same goal.

Amaral et al. (2020) analyze the relationship between co-changes and defect density (modifications in the same file) and merge conflicts with the propensity for commits that introduce bugs. The study was conducted based on 29 Apache Java projects. They argue that in many studies, these relationships are mentioned, but the research shows that merge commits are not more prone to introducing bugs than regular commits. Additionally, they point out that co-changes do not show a considerable interdependence relationship with code defect density.

Meng et al. (2021) employ Convolutional Neural Networks (CNN) to classify commits based on the modifications recorded in the commits and their relationships. In the reported study, the CNN method is applied to five open-source Java repositories; categorizing commits into bug-fix commits, feature insertions, and others (those that do not fit into the previous two classes). The authors compare the performance of the developed classifier with classifiers that use commit comments. Despite not making classifications into more detailed classes, only into the three mentioned, the CClassifier outperforms in performance.

The cited works aim, among other things, to obtain bug-fix commits for some purpose, whether associating them with complementary commits or predicting code issues. The present study uses commit comments for clustering and discovering activities performed, hoping to find the fixed activity among the discoveries. The analysis and discovery of patterns among commit comments are distinctive, as sentiment analysis within commits is used, which is not employed in other works.

Another point is that the present study performs commit clustering using clustering techniques, unlike the cited works, as most use classifiers for this activity with pre-labeled data.

3 THEORETICAL FRAMEWORK

In this section, the theoretical frameworks necessary for the development of this project are presented. Given that it involves more than one area (software repository, mining, data analysis) and their technologies, the theoretical framework is organized into subsections: Software Repository Mining (SRM), Data Mining Process, and Sentiment Analysis.

[2]https://github.com/BiancaPuertaRocha/Explore PopularRepos

3.1 Software Repository Mining (SRM)

Software Repository Mining (SRM) is a research area that supports the improvement of the software development process, maintenance, and prediction of tasks to be carried out in the project through the data made available by the platform that ships the repository(Keivanloo et al., 2012).

Therefore, SRM is the application of the concepts of data mining, discovery of patterns and/or associative rules in a software repository that contains historical records (metadata) of project modifications and public records about project collaborators.

For Cosentino et al. (2017), GitHub is the best platform to obtain data on activity in repositories as it provides public data for analysis of activities, which can lead to understanding the dynamics of software development communities and, therefore, has been used is the target of many studies and analyzes of metadata in research on online project management and software development.

3.2 Data Mining Process

The primary **pre-processing** goal is to make the data more suitable for data mining. It is a set of tasks to be performed on the dataset, but not all tasks are mandatory for every dataset. Thus, the database must be analyzed, and the necessary tasks should be defined (Tan et al., 2019).

Generally, the pre-processing transformations involve selecting or transforming the data. For this purpose, the following tasks are proposed (Tan et al., 2019): Aggregation, Sampling, Dimensionality reduction, Feature Subset Selecion, Discretization and binarization and Variable transformation.

For text databases, some concerns should be considered when preparing the dataset. Hickman et al. (2022) explain the pre-processing techniques and their benefits in data mining: (1) Lowercasing: This involves converting all letters to lowercase to ensure consistency and avoid treating variations of the same word with different letter cases as distinct. For instance, ”Translation” and ”translation” would be treated as the same word (2) Removal of Non-alphabetic Characters: This step eliminates all punctuation and non-alphabetic characters. It is common to remove only numbers and punctuation, preserving other characters if needed for specific analyses (3) Stemming: Removing word affixes to retain only the word's root. For instance, ”happiness,” ”happy,” and ”happily” might all stem from the root ”happi”. Emoticon Recognition, Lemmatization, Spell Correction, and Handling Negation were not used.

Subsequently, Feature Selection must be performed. This selection is a challenging task before mining; it requires processing words and assigning weights to them that represent their importance. Among several possible techniques is TF-IDF, which was employed in the study. This technique allows assigning weights to words based on their frequency and rarity in the dataset. Thus, if a word is ubiquitous, it is only considered important if it has sufficient weight to differentiate records. TF represents the frequency of the word in the document, while IDF is the inverse of the word's frequency in the same document (Qaiser and Ali, 2018).

Inside data mining, there is a specific concept of text mining. The text mining techniques allow knowledge extraction in non-structured data (Allahyari et al., 2017). After all the pre-process, considering the rules of the domain to which the data belongs and the question one aims to answer, it is possible to develop models capable of classifying or grouping collected and treated text data in categories.

When using documents, the intention is to group documents with the same subject, such as emails, articles, and text documents. When using paragraphs or sentences, the analysis involves sentences from different documents from the same source, for example, social media posts and comments. Finally, words associated with the same theme are grouped when analyzing words or terms.

For the **model construction**, the most common types of algorithms for clustering text data are: Hierarchical, partitioning, and density-based. Among them, the density-based was chosen to build the model in this paper, using the DBSCAN algorithm.

The decision to use the DBSCAN was moved by its capacity for automatic cluster number detection – a fundamental characteristic because the number of clusters in the dataset used is unknown. The DBSCAN can determine this number based on the data density, providing an effective solution that saves time and effort in defining the number of clusters.

In addition, the DBSCAN can also handle clusters of different shapes and sizes, an important characteristic of the studied dataset. Unlike other clustering algorithms, which assume clusters with uniform geometries, DBSCAN has flexibility when identifying clusters with different shapes, whether compact or more elongated. This flexibility allows for better identification of the clusters under analysis (Khan et al., 2014).

DBSCAN's ability to not be sensitive to the presence of outliers also played a crucial role in the decision to adopt it. The studied dataset contains outlier data points from most data, and DBSCAN can deal with these outliers effectively. These points were classified as ”noise” and excluded from the clusters, avoiding distortions in forming clusters and allowing the identified clusters to represent actual data patterns (Khan et al., 2014).

Considering the study of the set presented in Section 4, the characteristics mentioned were very important, with clusters of elongated shapes, non-uniform sizes, and outliers present.

After model construction, **post-processing** is done to visualize and interpret the patterns found after data mining, which has to be done before the integration with any system that will make available the discoveries made by the model developed (Tan et al., 2019). In this phase, a measurement of the model's quality and performance should also be conducted, and a thorough analysis of the grouped data should be conducted to identify patterns within the clusters.

To validate the developed model, various evaluation metrics, both internal and external, are employed. External evaluation occurs only when there is a predefined definition of clusters, enabling an assessment of the agreement between the groupings created by the model and the original clusters.

In internal evaluation, the relationship within and between clusters is studied. For this purpose, specific metrics are used, including the silhouette index, Davies-Bouldin index, and Calinski-Harabasz index. These indices are employed to assess the model compared to others and can also be used for refining hyperparameters to improve the model.

For the analysis of the created groups, this work employs the following measures: visualization of clusters through PCA (Principal Component Analysis) and analysis of keyword patterns within the clusters.

The **visualization of clusters through PCA** (Principal Component Analysis) enables the representation of the dataset in two dimensions, facilitating analysis of cluster separation. For that, it is necessary to generate two principal components using the data from existing features (Tan et al., 2019). In textual data, this application involves vectorizing the set of words for all records - each word corresponds to a column, and TF-IDF is used to calculate the frequency in the text. Thus, PCA can be applied to these numerical data, summarizing all the characteristics of the generated vectors into just two components (Indasari and Tjahyanto, 2023).

By analyzing the **keywords of the clusters**, it is possible to examine the most frequent words in each cluster to understand the purpose of the text group and the semantics that can be detected in the present phrases. For this identification, arranging the words in order of importance within the clusters is necessary based on the frequency metric used.

3.3 Sentiment Analysis

A text register group's sentiment analysis is more than searching for positive or negative sentiments. With this technique, it is possible to perform some analysis, called ”tasks” (Wankhade et al., 2022): Sentiments classification, Subjectivity classification, Opinion summary, Opinion retrieval, and Sarcasm and irony. In this paper, the task of Sentiments classification is performed. It considers the sentiment of those who wrote the object of study, categorizing them as negative, positive, and neutral.

Machine learning can be used to categorize objects into classes of sentiments using natural language processing techniques and computational text analysis. The categorization model can be built in a supervised or unsupervised manner. In supervised models, it is necessary to have a dataset with their labels to be used to train and validate the model. In unsupervised methods, object grouping occurs by analyzing prior knowledge bases of words, language, and ontologies that were previously constructed to assist in sentiment analysis (Wankhade et al., 2022).

4 METHODOLOGY

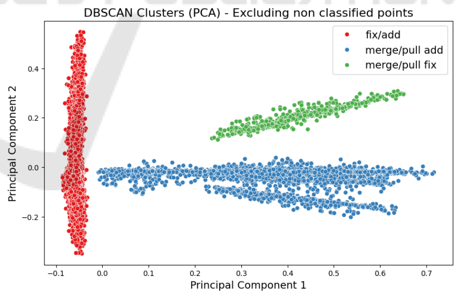

Figure 1: Clustering using DBSCAN.

Data Acquisition. The data was collected using Google's BigQuery tool, which allows SQL queries to GitHub's public dataset regarding open source repositories. First, twenty relevant repositories were chosen from five languages with the most popular repositories available on GitHub. Then, using the list of repositories, all commits from the selected repositories were consulted. According to their use on GitHub, from the most to the least popular, the five most popular languages are JavaScript, Ruby, Python, PHP, and Java. The list of languages, projects, and commits is also available in the author's GitHub repository.

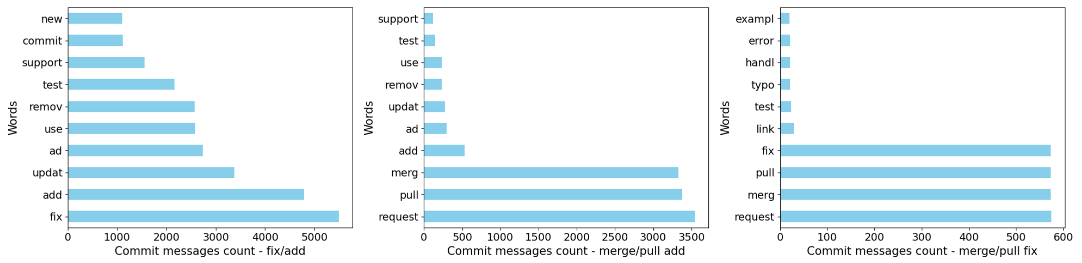

Figure 2: Sentiments in each cluster.

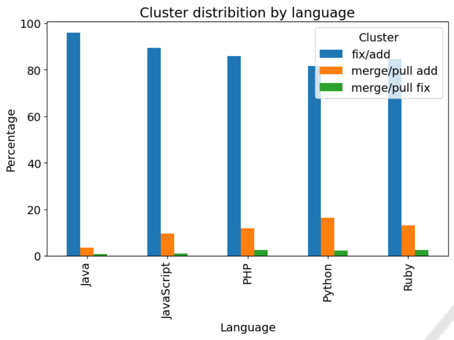

Figure 3: Cluster distribution by language.

Pre-Processing. Pre-processing consists of preparing data for data mining, whether for classification, clustering, or regression techniques. The first step was removing empty values as the other pre-processing steps cannot occur on empty values – and are also not interesting for analysis. Removal can only happen if the number of records with empty values is relatively small compared to the total, thus not interfering with the result and still allowing data mining with relevant and representative data. Then, according to the techniques discussed in Section 3, it was necessary to put all words in lowercase letters so that they are not considered different just because they are written in capital letters.

In this process, so-called stopwords are also unnecessary for the analysis. Therefore, Python's natural language processing library, NLTK (Natural Language Toolkit), was used to remove stopwords. The necessary next step was removing non-alphabetic characters because, when carrying out sentiment analysis, these characters tend to get in the way as they are not words with intrinsic sentiments. The NLTK library was also used for this processing.

Subsequently, it is necessary to remove no English words so that these words are not considered when analyzing sentiments and clustering. Stemming was applied to simplify the text by placing words in their roots, which makes it easier to manipulate large datasets and saves computational resources by reducing the words in the dataset. Records considered outliers were removed according to the length of the sentences – sentences with discrepant lengths. The thresholds for removal were calculated using the quartiles related to the size of the sentences.

Before applying the grouping technique through clustering, we perform PCA calculation on the dataset based on the TF-IDF related to the message column. This step aims to reduce dimensionality since vectorization with TF-IDF generates a column for each word, leading to a high set dimensionality.

By assigning weights to words based on their frequency in the document and the corpus, TF-IDF helps to diminish the weight of common words that may provide little information about the content of the message. This reduces noise in the clusters, enabling them to focus on more meaningful words.

The need to reduce dimensionality is justified by the fact that each vocabulary word becomes a dimension in the data set. As the number of dimensions increases, the complexity of the feature space grows exponentially, which can lead to problems such as the "curse of dimensionality", which affects effectiveness, particularly in density algorithms.

Model Construction. Three clustering algorithms were considered to build the clustering model: partitioning, hierarchical, and density-based. Density was the most suited to the data set studied; therefore, DBSCAN, one of the most used algorithms in this group, was used.

The partitioning, hierarchical, and density-based algorithms represent core classes of clustering algorithms that address different aspects and challenges of clustering analysis. This means they offer a good coverage of available clustering techniques.

The DBSCAN algorithm relies on two main parameters: the minimum number of points inside the radius to form the cluster (min_samples) and the maximum distance between points considered belonging to the same cluster (epsilon - eps), using Euclidean distance. The proper choice of these parameters is important to obtain good clustering results.

DBSCAN was used, changing the parameter values to identify it as the ideal combination. The goal is to determine the parameters that would result in the

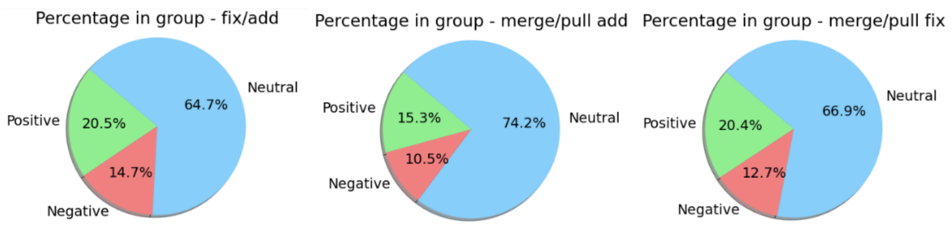

Figure 4: Frequent words inside clusters.

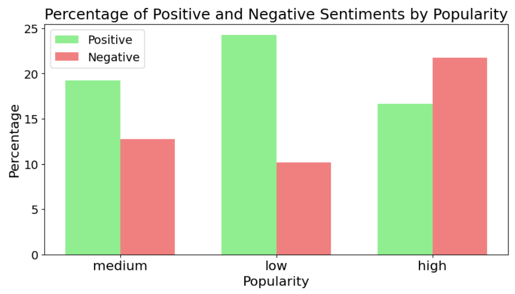

Figure 5: Sentiments in software repository according to popularity.

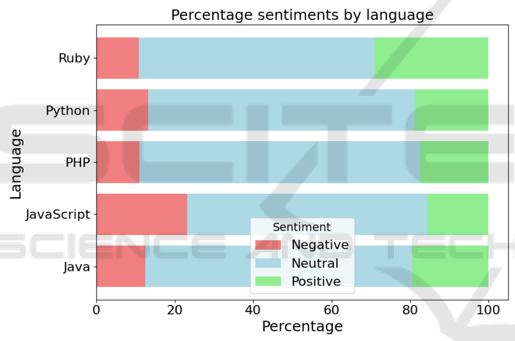

Figure 6: Sentiment in each programming language.

best clustering model for this dataset. The almost perfect choice of eps and min_samples was determined based on the best silhouette value because this metric evaluates the cohesion and separation of clusters. This procedure allowed the identification of parameters that generated the strong cluster structure to the dataset (Scitovski and Sabo, 2020).

The set of test hyperparameters was chosen based on manual testing, modifying the parameters until a range of data that best grouped the data points was identified. Thus, applying the described technique to refine the chosen hyperparameters was possible.

By applying PCA, the data is transformed into a smaller space, preserving the most relevant information. Then, the DBSCAN clustering algorithm was applied to this lower-dimensional space along with the hyperparameters found in the analysis, facilitating the identification of clustering patterns and improving the algorithm's effectiveness. Figure 1 shows the groups identified after PCA, removing data considered as outliers that do not belong to any group.

Post-Processing. It is essential to employ index analysis to improve and compare algorithms. However, data visualization is equally crucial in observing and evaluating the coherence of the clusters formed. Therefore, the clusters were visualized with each hyperparameter change to validate the model used for comparison with the other methods and the previous execution. Also, a visual exploration of the groups identified by the clustering model was carried out. The most frequently used words and sentiments expressed in each group were studied for this work.

Figure 2 shows that the defined groups have some more frequent words. The difference between the last two groups seems like a slight difference. However, it makes sense because in the merge/pull add group, merges, and pull requests the elements have a greater tendency to be updates, additions, or removals. The elements in the merge/pull fix group tend to be fixes (adjustments), errors, and tests. The fix/add group is very well defined, with simple commits to add, edit, or adjust features.

Finally, after carrying out the data grouping and clustering process, a crucial stage of analysis is carried out, which is sentiment analysis. For this, the *TextBlob* library in Python, a tool designed to evaluate the polarity and subjectivity of texts, was used. The *TextBlob* library simplifies the sentiment analysis process, allowing the evaluation of the polarity of the text, which can be classified as positive, negative, or neutral, and also of subjectivity, which indicates how subjective or objective the text is. These metrics are important for understanding the attitude expressed in a text and categorizing information meaningfully.

5 RESULT ANALYSIS

we have answered the formulated research questions in the following.

RQ1. Do programming languages influence the distribution of group proportions? Figure 3

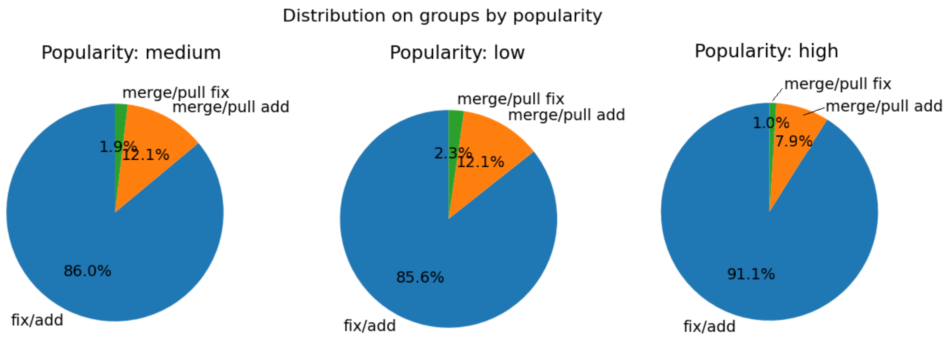

Figure 7: Repository groups in different popularities.

shows that the proportions of groups are similar in all programming languages in the dataset. However, merge/pull groups appear much less in Java repositories. A further study could explore why Java has more direct commits than merge/pull commits.

RQ2. Does the type of development activity influence the sentiments expressed in comments? The analysis in Figure 4 concludes that sentiments are predominantly neutral. However, in more direct activities, such as the fix/add group, sentiments are generally more positive than in other merge/pull activities, although surprisingly, negative sentiments stand out in merge/pull add (addition activities).

RQ3. Do sentiments expressed in commit comments reflect the project's popularity among developers? According to Figure 5, negative comments stand out in more popular repositories. Thus, they do not reflect what is expected of more popular repositories, which should have fewer errors and negative comments. This event can be studied more deeply later, as more popular repositories may have more development time and, consequently, more adjustments and errors due to the introduction of new features.

RQ4. Do the most used programming languages influence the sentiments expressed in comments? Figure 6 shows the number of commits with negative comments, which is prominent in JavaScript projects. Future studies can explore the cause of this phenomenon, possibly related to the high occurrence of code smells, which are closely linked to code errors (Johannes et al., 2019), resulting in more negative comments about adjustments or errors.

RQ5. Do the most recurring activities within the project have any relation to repository popularity? Figure 7 shows the percentage of each activity (group) found within repositories of three popularity levels. However, no percentage stands out enough to identify a difference pattern between them. Therefore, the studied dataset has no pattern related to activities and popularity.

6 CONCLUSIONS

The applied clustering model defined the activity groups, through which three main activities were discovered within the studied dataset. Subsequently, sentiment analysis was applied within these groups to uncover the sentiments expressed in committed comments while performing these activities. A graph was also generated to analyze the sentiments related to language groups and repository popularity, data obtained from the BigQuery collection.

The research questions help the project manager understand the developers' feelings for the most recurring activities and what influences those feelings. This understanding can be of great value for improving team management and function delegation by the manager. From a second perspective, those answers might be helpful to the team members, helping to avoid mistakes that degrade internal software quality, considering that negative comments express how badly a programmer found the source code and its issues to fix. It is not a long shot to consider the diversity of programming experience of those who modify the source code as a factor that introduces problems and the negative reaction of those tasked with fixing them. Although we do not present an analysis for each root word after stemming, it is clear that negative ones are greater in high-popularity repositories.

An important point, posing a threat to the conclusions and requiring exploration by the interested community, is the data acquisition for this study. The challenge arises from the limitation of requests in the GitHub API and the need for this data in the public GitHub database on BigQuery. However, other studies have faced similar problems, with considerable results (de Lima et al., 2022; Nguyen et al., 2023).

ACKNOWLEDGMENTS

This study was financed in part by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior – Brasil (CAPES).

REFERENCES

Allahyari, M., Pouriyeh, S., Assefi, M., Safaei, S., Trippe, E. D., Gutierrez, J. B., and Kochut, K. (2017). A brief survey of text mining: Classification, clustering and extraction techniques. *arXiv preprint arXiv:1707.02919*.

Amaral, L. et al. (2020). How (not) to find bugs: The interplay between merge conflicts, co-changes, and bugs. In *2020 IEEE Int. Conf. on Software Maintenance and Evolution (ICSME)*, pages 441–452.

Cosentino, V., Izquierdo, J.-L. C., and Cabot, J. (2017). A systematic mapping study of software development with github. *IEEE access*, 5:7173–7192.

de Lima, B. S., Garcia, R. E., and Eler, D. M. (2022). Toward prioritization of self-admitted technical debt: an approach to support decision to payment. *Software Quality J.*, pages 1–27. https://doi.org/10.1007/s11219-021-09578-7.

Hasan, K., Macedo, M., Tian, Y., Adams, B., and Ding, S. (2023). Understanding the time to first response in github pull requests. In *2023 IEEE/ACM 20th Int. Conf. on Mining Software Repositories (MSR)*, pages 1–11, Los Alamitos, CA, USA. IEEE Computer Society.

Hickman, L., Thapa, S., Tay, L., Cao, M., and Srinivasan, P. (2022). Text preprocessing for text mining in organizational research: Review and recommendations. *Organizational Research Methods*, 25(1):114–146.

Indasari, S. S. and Tjahyanto, A. (2023). Automatic categorization of multi marketplace fmcgs products using tf-idf and pca features. *Jurnal Sisfokom (Sistem Informasi dan Komputer)*, 12(2):198–204.

Ji, T., Pan, J., Chen, L., and Mao, X. (2018). Identifying supplementary bug-fix commits. In *2018 IEEE 42nd Annual Computer Software and Applications Conf. (COMPSAC)*, pages 184–193.

Johannes, D., Khomh, F., and Antoniol, G. (2019). A large-scale empirical study of code smells in javascript projects. *Software Quality J.*, 27:1271–1314.

Keivanloo, I., Forbes, C., Hmood, A., Erfani, M., Neal, C., Peristerakis, G., and Rilling, J. (2012). A linked data platform for mining software repositories. In *2012 9th IEEE Working Conf. on Mining Software Repositories (MSR)*, pages 32–35, Zurich, Switzerland. IEEE.

Khan, K., Rehman, S. U., Aziz, K., Fong, S., and Sarasvady, S. (2014). Dbscan: Past, present and future. In *The fifth Int. Conf. on the applications of digital information and web technologies (ICADIWT 2014)*, pages 232–238. IEEE.

Kinsman, T., Wessel, M., Gerosa, M. A., and Treude, C. (2021). How do software developers use github actions to automate their workflows? In *2021 IEEE/ACM 18th Int. Conf. on Mining Software Repositories (MSR)*, pages 420–431, Los Alamitos, CA, USA. IEEE Computer Society.

Meng, N., Jiang, Z., and Zhong, H. (2021). Classifying code commits with convolutional neural networks. In *2021 Int. Joint Conf. on Neural Networks (IJCNN)*, pages 1–8. IEEE.

Nguyen, P. T., Rubei, R., Rocco, J. D., Sipio, C. D., Ruscio, D. D., and Penta, M. D. (2023). Dealing with popularity bias in recommender systems for third-party libraries: How far are we? In *2023 IEEE/ACM 20th Int. Conf. on Mining Software Repositories (MSR)*, pages 12–24, Los Alamitos, CA, USA. IEEE Computer Society.

Qaiser, S. and Ali, R. (2018). Text mining: use of tf-idf to examine the relevance of words to documents. *Int. J. of Computer Applications*, 181(1):25–29.

Scitovski, R. and Sabo, K. (2020). Dbscan-like clustering method for various data densities. *Pattern Analysis and Applications*, 23(2):541–554.

Tan, P.-N., Steinbach, M., and Kumar, V. (2019). *Introduction to Data Mining*. Pearson, Boston, MA, 2nd edition.

Wankhade, M., Rao, A. C. S., and Kulkarni, C. (2022). A survey on sentiment analysis methods, applications, and challenges. *Artificial Intelligence Review*, 55(7):5731–5780.

Zafar, S., Malik, M. Z., and Walia, G. S. (2019). Towards standardizing and improving classification of bug-fix commits. In *2019 ACM/IEEE Int. Symp. on Empirical Software Engineering and Measurement (ESEM)*, pages 1–6.