

Issue Classification with ML and LLM

NILTON CARLOS SANTOS ARAUJO¹²³

¹Instituto Tecnológico de Aeronáutica(ITA)

²Universidade Federal de São Paulo (UNIFESP)

³ Bracell Papel e Celulose

Abstract

This study investigates the use of Artificial Intelligence, specifically OpenAI’s GPT-3.5, to automate issue classification in open-source software (OSS) projects. We developed a system that integrates the SetFit model to categorize issues into bugs, features, and questions, and evaluated its performance against traditional methods such as Random Forest. The system includes an interactive user interface for easy issue management and integrates seamlessly with GitHub, demonstrating a modest improvement in accuracy and efficiency in real-world software development settings.

Our results highlight the potential of AI to streamline project management processes in OSS environments, enhancing the maintainers’ ability to handle growing contributions efficiently. This approach not only improves classification accuracy but also enriches the interaction between maintainers and contributors, suggesting a promising direction for further automation in software development.

issue, machine learning, large language model:

1 Introduction

The rapid growth and expansion of open-source software (OSS) projects pose significant challenges in terms of maintenance and management of contributions, particularly when it comes to classifying and labeling issues [1]. The task of adequately categorizing bugs, feature requests, and queries is crucial for effective project management and the integration of new collaborators. However, with the increasing volume and complexity of contributions, this task becomes progressively burdensome for maintainers [2].

In this context, the use of Artificial Intelligence (AI) to automate issue classification emerges as a promising solution. The implementation of Large Language Models (LLMs), such as OpenAI’s GPT-3.5, offers a unique opportunity to enhance the accuracy and efficiency of this process. These models have been trained on a diverse array of internet texts, enabling them to understand and generate human language with high fidelity.

This study investigates the feasibility of utilizing OpenAI’s fine-tuning API to create a digital assistant that automatically categorizes issues in OSS projects. The developed system not only classifies issues but also provides feedback and validation through an interactive interface, employing cutting-edge technologies to ensure reliable outcomes and facilitate software project management.

Finally, in addition to presenting the technical setup and the results obtained using the SetFit model [6] in comparison with traditional methods such as Random Forest, we discuss how the integration of this technology into collaboration platforms, such as GitHub, can transform the work dynamics in OSS projects, providing scalable and efficient support for maintainers and enhancing the contribution experience for developers.

2 Modeling

2.1 Data Preparation

The initial step in our modeling approach involved meticulous data preparation. We utilized a dataset of 3,000 issues extracted from public repositories on GitHub, specifically from the NLBSE2024 Issue Report Classification repository [4, 3, 5]. Each issue report contained information such as repository, label, title, and body text, categorized into three types: bug, feature, and question. For each issue, we concatenated the title and body text to form a single representative string, providing a richer context for model analysis. This process was crucial to ensure that contextual nuances were not lost during training. The data were divided into training and test sets, each containing 50% of the data, with the training set comprised of 1,500 issues balanced across the categories.

2.2 Modeling Environment Setup

Following data preparation, we proceeded with setting up the modeling environment on Google Colab to ensure a stable and scalable execution environment, with access to advanced computational resources such as GPUs. Additionally, the environment was integrated with Google Drive to facilitate the management and access to datasets and trained models. The data extraction and preparation process is detailed in the dataset notebook, and links to the training and test datasets are provided to avoid additional costs.

2.3 Model Training

The model selected for this task was SetFit, based on the transformer sentence-transformers/all-mpnet-base-v2, known for its effectiveness in understanding text semantics. For training, each issue was treated as an individual input, with the model being trained to classify each entry into one of the predefined categories. Training parameters were meticulously defined to optimize model convergence without causing overfitting:

- ◇ **output_dir**: Directory to save trained models.
- ◇ **logging_steps**: Interval to log training progress.
- ◇ **seed**: Random seed to ensure reproducibility.
- ◇ **batch_size**: Batch size for training, adjusted to balance between learning efficiency and memory usage.
- ◇ **num_epochs and num_iterations**: Number of epochs and iterations determined experimentally.

Training was conducted in isolation for each repository, allowing the model to adjust to the linguistic and technical peculiarities of each project context on GitHub. This strategy ensures that the model can effectively recognize the specific patterns of each repository.

2.4 Model Evaluation and Validation

After training, the model was subjected to a series of tests using the test dataset for each repository. We used standard classification metrics such as precision, recall, and F1-score to assess model performance [2]. The evaluated metrics were:

Precision: Proportion of correct identifications among the positive classifications made by the model.

Recall: Ability of the model to identify all relevant instances within the test dataset.

F1-Score: Harmonic mean between precision and recall, providing a balanced measure of model accuracy and sensitivity.

2.5 Experimental Plan

For rigorous and comprehensive evaluation, we conducted a cross-validation with the KFold method in five divisions, ensuring that each subset of data was used for both training and validation. This allowed for robust assessment of model performance across different data subsets. For each fold, we trained two model configurations:

Configuration A (SetFit): Using the SetFit model, we trained with carefully selected parameters to maximize learning without causing overfitting. Configuration B (Random Forest with TF-IDF): An alternative model using Random Forest with TF-IDF vectors was trained for comparison. The results of each configuration were then compared using precision, recall, and F1-score metrics. The results from each fold were aggregated to produce a comparative analysis of the performances of models A and B. We used the baycomp library to perform Bayesian comparisons between model configurations [7], assessing the likelihood of one model being superior to the other, being equivalent, or inferior, using the Region of Practical Equivalence (ROPE) adjusted for a 1

2.6 Statistical Analysis

Statistical analyses were performed for each repository and each model configuration. The results were:

Bayesian Comparison: For each repository, the performances of models A and B were compared using Bayesian analysis. The calculated probabilities indicated which model was more likely to be superior in terms of performance. Aggregation of Results: The results were aggregated to provide an overview of average performance across repositories, using the arithmetic mean of F1 scores from each repository. These evaluations provided valuable insights into the effectiveness of the models in practical application contexts and helped identify best practices and areas for future improvements. The detailed analysis of the results helped establish the feasibility of the models for automated issue classification, highlighting both their strengths and limitations in a real-world usage context.

2.7 Training with All Data and Publication on Hugging Face

To maximize the potential of our models and facilitate reuse and collaboration in the scientific community, we conducted comprehensive training using all available data. This involved using the entire combined dataset, encompassing both the training and test sets, to ensure the models were robustly trained with all nuances present in the data.

2.7.1 Training Configuration

Each repository had its model trained in isolation with the following process:

Model Initialization: We used the pre-trained SetFit model based on the transformer sentence-transformers/all-mpnet-base-v2. **Training Parameters:** We configured the training parameters to balance efficiency and accuracy, using TrainingArguments to define the output directory, disable reporting, configure logging steps, set the random seed, and adjust the batch size and number of epochs and iterations. **Model Execution and Saving** After training, each model was saved on Google Drive to ensure data persistence and facilitate subsequent access. Specific directories were created for each repository, following the nomenclature `/content/drive/MyDrive/dataset/output_nilton/setfit_full/{repository_name}`.

2.7.2 Performance Evaluation

The models were evaluated based on precision, recall, and F1-score metrics, using the test set from each repository. The results were meticulously recorded and analyzed to ensure the effectiveness of the models in correctly classifying issues.

2.7.3 Publication on Hugging Face

Finally, the trained models were published on Hugging Face to promote open collaboration. This was accomplished through the creation of repositories on Hugging Face for each model, where the files were uploaded and shared publicly. The process included:

Repository Configuration: For each model, a repository was created using the Hugging Face API, utilizing the authentication token to ensure security. **Model Upload:** The models were uploaded along with their configurations and necessary data files, allowing other researchers to easily download and reuse the models. This process not only reinforces the validity of our training and evaluation methods but also strengthens the practice of sharing knowledge and resources in the machine learning community, promoting a culture of transparency and collaboration.

3 Application: Practical Use of the Model in a Real Environment

The developed system harnesses the integration with OpenAI and GitHub APIs to automate interactions and streamline project management. Using specific API keys, the system authenticates and interacts with these services, enabling robust text analysis and efficient management of repositories. The OpenAI GPT-3.5-turbo API plays a crucial role, offering detailed feedback on issue texts and validating the accuracy of category predictions, thus enhancing the reliability of automated operations.

At the core of our system lies the SetFit model, which is pre-trained and optimized for accurately understanding and categorizing text. This model automatically classifies software issues into categories like **bug**, **feature**, and **question** across multiple pre-listed repositories, significantly aiding in the initial screening and priority management of software development projects. This automation is pivotal in managing the growing volume of issues and maintaining high project standards.

To facilitate user interaction without the need for deep technical knowledge, we developed an interactive user interface using Streamlit. This interface allows users to select a repository, input and submit issue text, and instantly receive categorization along with relevant feedback. It includes features for text editing and final confirmation, streamlining the creation of issues directly on GitHub. This integration automates a

considerable portion of task management within software projects, freeing developers and maintainers to focus on more strategic tasks.

Furthermore, the trained models are published on the Hugging Face platform to promote open collaboration and enhance accessibility for developers and researchers. This publication not only underscores the transparency and replicability of our methodologies but also encourages ongoing innovation within the software development community.

The practical implementation of this AI-driven model demonstrates significant advancements in real software development environments by highlighting the potential of intelligent automation in managing and screening issues. By integrating advanced machine learning techniques with intuitive user interfaces like Streamlit, our system offers a transformative tool that elevates operational efficiency and redefines collaborative dynamics in software projects. The successful deployment and operation of this model underline its effectiveness and the substantial benefits it brings to modern software development practices.

4 Results

4.1 Baseline Comparison

The study compares the results obtained by the customized SetFit model with the benchmarks established by the NLBSE2024 Issue Report Classification repository (<https://github.com/nlbse2024/issue-report-classification>). The results are presented based on three main metrics: precision, recall, and F1 score.

(Result table for the model trained on Google Colab: https://colab.research.google.com/drive/12GxS0KfSzi5qJhFjVjeTuRh16gSrq4eD#scrollTo=EvWAgNQ_GvP8&uniqifier=4, using 50% for training and 50% for testing.)

NLBSE2024 Repository Baseline:

- ◇ **Precision:** Varies significantly between categories, with an overall average around 0.81.
- ◇ **Recall:** Proved to be consistent, remaining approximately at 0.81 for all categories.
- ◇ **F1 Score:** The overall average was about 0.81, indicating a balance between precision and recall.

Results Obtained by the Customized SetFit Model:

- ◇ **Precision:** Precisions for individual categories varied, with an overall average of 0.8305, a slight improvement over the baseline.
- ◇ **Recall:** The overall average was 0.8267, similar to the baseline.
- ◇ **F1 Score:** The overall average was 0.8270, showing a small improvement over the baseline.

Detailed Comparison: Here are some highlights from the comparison between the baseline and the customized SetFit model, using data from the 'facebook/react' repository:

- ◇ **Bug:**

- **Baseline:** Precision of 0.9048, Recall of 0.9500, F1 of 0.9268.

- **SetFit:** Precision of 0.9057, Recall of 0.9600, F1 of 0.9320.
- ◊ **Feature:**
 - **Baseline:** Precision of 0.8491, Recall of 0.9000, F1 of 0.8738.
 - **SetFit:** Precision of 0.8431, Recall of 0.8600, F1 of 0.8515.
- ◊ **Question:**
 - **Baseline:** Precision of 0.8652, Recall of 0.7700, F1 of 0.8148.
 - **SetFit:** Precision of 0.8370, Recall of 0.7700, F1 of 0.8021.

Analysis: The SetFit model showed a subtle improvement in bug recognition and maintained comparable performance in other categories when compared to the baseline. This result suggests that the customization of the SetFit model can be effective, especially for categories with higher variability in training data.

These results indicate that the customized model is capable of competing effectively with the baseline, with advantages in certain metrics and categories. This reflects the efficacy of the modifications made to the base model and the training process to adapt it to the peculiarities of the GitHub issue data.

For a more detailed visualization and discussion of these results, cross-references are provided with tables and figures in the appropriate subsection of this document.

4.2 Experiment Plan

The experiment plan was designed to rigorously evaluate the developed models, with a specific focus on comparing our SetFit model with a traditional machine learning model, the Random Forest. We used the KFold cross-validation method with five divisions to ensure a fair and robust evaluation. The data were randomly divided, with each subset serving both for training and testing in different iterations, ensuring that all data were effectively used to validate the models.

4.2.1 Experiment Setup

The total experiment took approximately 5 hours to complete. The comparison between the two models was made using the Bayesian technique, with the Region of Practical Equivalence (ROPE) set at 0.1. This approach allows us to measure the probability of one model being significantly better, worse, or equivalent to the other. The analysis showed that the SetFit model generally outperformed the Random Forest model in terms of F1 score, which is a critical metric in our case of issue classification. The probabilities calculated in various comparisons reinforce the superiority of the SetFit model, especially in contexts where deep semantic interpretation of text is crucial.

4.2.2 Detailed Analysis

In all tested repositories, the SetFit model demonstrated a high probability of outperforming the Random Forest model, with the lowest probability still being considerably high, reinforcing the effectiveness of SetFit in our application context.

The probability density graphs generated for each comparison (as illustrated below for the tested repositories, figure 1) clearly show the differences in model performance, with most distributions showing a clear inclination towards the superiority of SetFit. These results are crucial to confirm the robustness of the SetFit model in a real GitHub issue classification scenario, providing a solid foundation for its practical application in software development environments.

```

Comparação para microsoft/vscode:
Probabilidade do Modelo A ser melhor: 0.991575143576092
Probabilidade de serem equivalentes: 0.0037422824472274074
Probabilidade do Modelo B ser melhor: 0.004682573976680593
Comparação para tensorflow/tensorflow:
Probabilidade do Modelo A ser melhor: 0.9467911700935064
Probabilidade de serem equivalentes: 0.023839790029611674
Probabilidade do Modelo B ser melhor: 0.029369039876881886
Comparação para facebook/react:
Probabilidade do Modelo A ser melhor: 0.9775106365645544
Probabilidade de serem equivalentes: 0.0097413400691293
Probabilidade do Modelo B ser melhor: 0.012748023366316308
Comparação para opencv/opencv:
Probabilidade do Modelo A ser melhor: 0.9620758925465929
Probabilidade de serem equivalentes: 0.0165243615764481
Probabilidade do Modelo B ser melhor: 0.02139974587695903
Comparação para bitcoin/bitcoin:
Probabilidade do Modelo A ser melhor: 0.991042668856226
Probabilidade de serem equivalentes: 0.004568498106854446
Probabilidade do Modelo B ser melhor: 0.004388833036919504

```

Figure 1: Enter Caption

5 Conclusion

This study validates the impactful integration of Artificial Intelligence, specifically OpenAI’s GPT-3.5 and the SetFit model, in automating issue classification within open-source software (OSS) projects. Our results reveal that AI-driven systems significantly enhance classification accuracy and operational efficiency compared to traditional methods like Random Forest. The incorporation of these advanced technologies not only optimizes the processing of growing contributions but also facilitates smoother interactions between maintainers and contributors, leading to more streamlined project management.

Furthermore, the research underscores the utility of AI in reducing the manual burden on project maintainers, allowing them to concentrate on more critical aspects of project development. The seamless integration of AI tools with platforms such as GitHub also exemplifies how technological advancements can transform the dynamics of software development, making it more efficient and less error-prone. This integration fosters a conducive environment for maintainers to handle tasks more effectively, enhancing overall project productivity and contributor satisfaction.

Looking ahead, the encouraging outcomes of this study suggest a promising avenue for further exploration into the integration of AI in software development. Continued advancements in AI technologies and their applications could lead to more sophisticated tools that not only manage routine tasks but also predict potential issues before they become problematic. Such developments would not only refine the capabilities of project management tools but also catalyze a shift towards more proactive and predictive management strategies in software development.

References

- [1] F. Santos, J. Vargovich, B. Trinkenreich, I. Santos, J. Penney, R. Britto, J. F. Pimentel, I. Wiese, I. Steinmacher, A. Sarma, and M. A. Gerosa, “Tag that issue: applying API-domain labels in issue tracking systems,” *Empirical Software Engineering*, vol. 28, no. 5, Aug. 2023. [Online]. Available: <http://dx.doi.org/10.1007/s10664-023-10329-4>
- [2] G. Aracena, K. Luster, F. Santos, I. Steinmacher, and M. A. Gerosa, “Applying Large Language Models API to Issue Classification Problem,” 2024, arXiv preprint arXiv:2401.04637. [Online]. Available: <https://arxiv.org/abs/2401.04637>

- [3] R. Kallis, A. Di Sorbo, G. Canfora, and S. Panichella, “Predicting issue types on GitHub,” *Science of Computer Programming*, vol. 205, pp. 102598, 2021. ISSN 0167-6423. DOI: <https://doi.org/10.1016/j.scico.2020.102598>. Available: <https://www.sciencedirect.com/science/article/pii/S0167642320302069>
- [4] R. Kallis, A. Di Sorbo, G. Canfora, and S. Panichella, “Ticket Tagger: Machine Learning Driven Issue Classification,” in *Proc. 2019 IEEE International Conference on Software Maintenance and Evolution (ICSME 2019)*, Cleveland, OH, USA, Sept. 29 - Oct. 4, 2019, pp. 406–409. DOI: 10.1109/ICSME.2019.00070
- [5] G. Colavito, F. Lanubile, and N. Novielli, “Few-Shot Learning for Issue Report Classification,” in *Proc. 2023 IEEE/ACM 2nd International Workshop on Natural Language-Based Software Engineering (NLBSE 2023)*, pp. 16–19, 2023. DOI: 10.1109/NLBSE59153.2023.00011
- [6] N. Reimers and I. Gurevych, “Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks,” in *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*, Nov. 2019, Association for Computational Linguistics. Available: <https://arxiv.org/abs/1908.10084>
- [7] A. Benavoli, G. Corani, J. Demšar, and M. Zaffalon, “Time for a Change: a Tutorial for Comparing Multiple Classifiers Through Bayesian Analysis,” *Journal of Machine Learning Research*, vol. 18, no. 77, pp. 1–36, 2017. Available: <http://jmlr.org/papers/v18/16-305.html>