Conference Paper

Software Skills Identification: A Multi-Class Classification on Source Code Using Machine Learning

Dimitris Bamidis, Ilias Kalouptsoglou, Apostolos Ampatzoglou, Alexandros Chatzigeorgiou*

University of Macedonia, Thessaloniki, Greece.

* Corresponding Author Email: achat@uom.edu.gr

ABSTRACT

In the ever-evolving tech industry, accurately assessing the software skills of developers is critical for effective workforce management. This study presents a machine learning approach to classify software development knowledge through source code analysis, focusing on Java-based technologies. A dataset of several source code files from multiple domains of software development was compiled from public repositories and labeled for classification. The high performance achieved in this study, by applying transfer learning, underlines the suitability of pre-trained CodeBERT models for the classification of software skills.

The methodology combined both non-pretrained neural networks and pretrained models to enhance classification accuracy. Results validate the feasibility of using machine learning to identify developers' programming proficiencies, providing a foundation for sophisticated assessment tools. Future work aims to refine classification by incorporating functional task identification and commit-based analysis for a more comprehensive evaluation of coding skills. This study showcases the transformative potential of machine learning in streamlining developer assessments and advancing software engineering methodologies.

Keywords—*Machine learning, Supervised learning, Multi-class classification, Neural network, Transfer learning, Source code analysis.*

INTRODUCTION

In today's fast-paced tech industry, it has become increasingly difficult for companies to evaluate the skills of job applicants, leading to incorrect allocation of tasks and suboptimal hiring decisions. As a solution to this problem, this work utilized a machine learning-based model that can effectively classify the software knowledge of developers, by recognizing the different technologies and programming languages implemented by them, thus assisting companies in managing their workforce based on acquired skills. We collected data from various Java-based software technologies and employed machine learning techniques to classify each source code file. A pre-trained CodeBERT[1] model was implemented for the multi-class classification task and provided very high accuracy and precision. Based on previous work[2], we implemented source code analysis by applying Natural Language Processing (NLP) techniques. The resulting model can be used as an effective tool for assessing the software knowledge of developers.

METHODS

The methodology employed in this research consisted of several key steps to address the problem of multi-class classification of source code. The methodology pipeline is presented in Figure 1.

FIGURE 1. Mobile Virtual Patients App interface.

Experimental Environment

The experiments on source code classification were conducted using the Jupyter Notebook from Anaconda as a primary development environment. To accelerate the computations, we utilized NVIDIA's CUDA platform to parallelize computations on the graphics card, which had a significant impact, when compared to a CPU-only approach. In terms of libraries and frameworks, several essential tools were used. TensorFlow, an open-source machine learning framework, played a central role in building and training the neural network models for source code classification. To evaluate the performance of the models, the scikit-learn (sklearn) library was selected, as it provided various utilities for data preprocessing, model evaluation, and performance metrics calculation. By utilizing sklearn, we could assess the accuracy, precision, recall, F1-score, and confusion matrix of our source code classification models, enabling a comprehensive analysis of their effectiveness. Lastly, to enhance the capabilities of the models, we utilized the CodeBERT model from the Transformers library. Transformers is a powerful library for NLP tasks, including source code understanding and processing.[3] The pre-trained CodeBERT model allowed us to benefit from transfer learning[4], as it had been pre-trained on meaningful representations of source code from large scale code corpora.

Data selection

The data selection process played a crucial role in obtaining a representative dataset for source code classification. In the present research, we collected the necessary source code files from public GitHub repositories and selected multiple Java source code files that we considered representative of each one of the following classes of software technology. We used a total of 183 files for the training and validation process. The six classes selected were: 1) JDBC (Java Database Connectivity), 2) File handling, 3) Exception handling, 4) Unit testing, 5) GUI (Graphical User Interface), 6) Miscellaneous. As inputs to the ML models entire Java files were used, however the problem and the models themselves can be generalized to snippets of code, such as code commits during changes in a software repository. Thus, the files have been manually labeled regarding the programming Java concepts that they are mostly related to.

RESULTS

For evaluating the pre-trained CodeBERT model's performance in the multi-class classification of source code, we employed a set of appropriate evaluation metrics. These metrics include precision, recall, and F1 score. The model's precision for the current task of identifying

the technology of the source code files in JAVA, achieved 91%, the model's recall reached 90% and the F1 score achieved 90% (see Figure 2).

```
Precision:0.9132061354283576, Recall:0.9074074074074074, F1 score:0.905842151675485.
```

FIGURE 2. The evaluation metrics of the pre-trained CodeBERT model.

To get an insight into the model's performance and behavior during the training process, we also provide a plot with the model's training loss and validation loss metrics. The following plot serves as a diagnostic tool to assess the model's learning dynamics and generalization ability (see Figure 3).

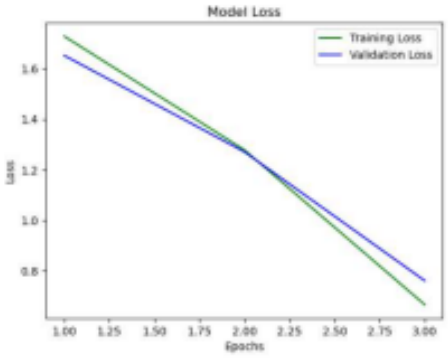

FIGURE 3. The model loss plot.

For a test case of 56 files from our dataset in which we performed multi-class classification, we created a confusion matrix (see Figures 4 and 5). The dataset contained 9 files from class "Exceptions", 12 files from class "File Handling", 10 files from class "GUI", 10 files from class "JDBC", 7 files from class "Unit Testing" and 8 files from class "Others".

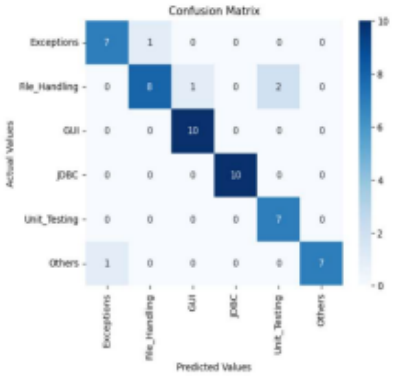

FIGURE 4. Classification's confusion matrix.

```
True Positives: [ 7  8 10 10  7  7]
False Positives: [1 1 1 0 2 0]
False Negatives: [1 3 0 0 0 1]
True Negatives: [45 42 43 44 45 46]
```

FIGURE 5. Matrix of TP, FP, FN,TN.

DISCUSSION

In this paper, we have managed to introduce a methodology for identifying software skills from source code using machine learning algorithms. Furthermore, this paper contributed to the field of software engineering by demonstrating the practical applicability of machine learning for software analysis and also to the understanding of software skills identification by investigating the impact of different features on the accuracy of the classification model.

Limitations

While this study demonstrates promising results and provides valuable insights into multi-class classification of source code, it is crucial to recognize the limitations stemming from the small dataset size, the context-specific evaluation metrics, and the potential constraints of transfer learning with a pre-trained model. By acknowledging these limitations and considering them in the interpretation of the findings, future research can build upon this work and advance the development of more robust and versatile code classification systems.

Future Extensions

Furthermore, beyond identifying different technologies in the code files, the next step could involve recognizing the specific tasks performed within the code. This would involve a more granular analysis to classify code based on the functionalities it serves, such as data manipulation, algorithm implementation, user interface development, or database management. By incorporating task identification, the classification system could provide deeper insights into developers' programming skills and aptitudes in different areas. Additionally, a source code analysis in commits from repositories could be introduced as an assessment tool. By integrating the commit analysis process, developers would gain valuable insights into the changes

introduced by the commits and obtain essential information about the source code. Future work could involve exploring machine learning approaches to automatically classify the nature and impact of the commits based on the analysis of source code.

CONCLUSION

Through the analysis, we have obtained valuable insights into the effectiveness of neural networks, the benefits of transfer learning using pre-trained models, and the potential for developing an assessment tool for developers. We exploited the power of transfer learning by employing the pre-trained CodeBERT model. This approach allowed us to capitalize on the vast amount of knowledge captured by the pre-trained model on a diverse range of source code tasks. By fine-tuning CodeBERT on our specific multi-class classification task, we were able to achieve impressive performance in terms of evaluation metrics, indicating the robustness and effectiveness of the transfer learning approach. The successful implementation of the multi-class classification task for recognizing different technologies in the Java programming language lays the foundation for the development of an assessment tool for developers.

REFERENCES

1. Feng, Z., Guo, D., Tang, D., et al. CodeBERT: A Pre-Trained Model for Programming and Natural Languages. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, 16–20 November 2020, pp. 1536–1547; Association for Computational Linguistics: Kerrville, TX, United States. https://doi.org/10.18653/v1/2020.findings-emnlp.139.
2. Kourtzanidis,S., Chatzigeorgiou, A., Ampatzoglou, A. RepoSkillMiner: identifying software expertise from GitHub repositories using natural language processing. In *Proceedings of the 35th IEEE/ACM International Conference on Automated Software Engineering (ASE '20)*, Melbourne, Australia, 21–25 September 2020, pp. 1353–1357; Association for Computing Machinery, New York, NY, United States. https://doi.org/10.1145/3324884.3415305.
3. Zhang, K., Li, G., Jin, Z. What does Trans former learn about source code? 2022, arXiv preprint. https://doi.org/10.48550/arXiv.2207.08466.
4. Sharma, T., Efstathiou, V., Louridas, P., et al. Code smell detection by deep direct-learning and transfer-learning. *J Syst Softw.* 2021;176:110936. https://doi.org/10.1016/j.jss.2021.110936.