Unveiling Hidden Patterns: Clustering Algorithms on C Code embedding

Spoorthi M, Richa Vivek Savant, Samarth Seshadri, Nakka Narmada and Peeta Basa Pati
Department of Computer Science and Engineering
Amrita School of Computing, Bengaluru
Amrita Vishwa Vidyapeetham, India
*Corresponding Author: bp_peeta@blr.amrita.edu, ORC ID: 0000-0003-2376-4591

***Abstract*—Automatic grading can help streamline assessment by using advanced algorithms to evaluate embedded C programs swiftly and objectively. This technology ensures standardized evaluations, providing quick and objective feedback on programming proficiency, enabling educators and employers to gauge participants' skills accurately. This paper proposes leveraging the application of machine learning clustering methods on embedding of C programs that have been examined by subject matter experts (SMEs) to uncover hidden patterns in the input vectors. A variety of clustering techniques are applied at the score level to capture the patterns in the data and analyze if they lie at par with the SMEs' evaluations. Machine learning models can at best capture clusters at the questions level, i.e., it can capture similarities amongst text, but cannot comprehend the logic behind it and classify the types of errors in code solutions. K-means clustering proves to provide the best compact clusters at a question level of the chosen dataset with biased data distribution in the markings.**

***Index Terms*—Automatic grading, embedding C programs, Machine learning, clustering, SMEs, similarities, score, Questions.**

I. Introduction

This study embarks on a pioneering journey into code pattern analysis and the utilization of machine learning techniques to develop automated scoring systems. The study goes beyond traditional code evaluation methodologies by delving into the understanding of code embeddings, which serve as complex and context-rich representations of source code context derived from sophisticated deep learning models [1]. Emphasizing the complex task of evaluating code quality and functionality, the focus extends to careful comparison with scores generated by clustering methods. By leveraging the capabilities of machine learning clustering techniques, the aim is to uncover subtle patterns in coding structures and code quality that are consistent with human acceptability, contributing to the evolving field of automated code evaluation, which helps ensure that assessments are consistent and that participants receive prompt, unbiased feedback on their programming skills [2]. This helps employers and educators make accurate assessments of the participants' abilities. This approach seeks to improve the understanding of how automated systems can augment human-level evaluation and offers insight into the potential of these techniques to revolutionize code evaluation processes across various software engineering applications.

Robust machine learning clustering techniques allow computers to recognize patterns and combine related data points into groups. In unsupervised learning, such as clustering, the algorithm looks for the underlying structure in the data without explicit direction. Because of their adaptability, clustering models are very useful for gleaning insightful information from complicated datasets, which facilitates decision-making in a variety of contexts. Clustering techniques have the potential to help produce more complex and precise data analysis as machine learning advances.

Clustering is a powerful machine learning technique for pattern analysis, which finds correlations, similarities, and underlying structures in a dataset. By grouping data points according to shared features, clustering algorithms make it possible to identify patterns that might not be immediately obvious. Furthermore, anomaly detection, in which departures from known patterns are noted as possible problems, makes extensive use of clustering [3].

The primary questions covered in this research are:

- How well does the embedding from pre-trained models capture the input code?
- What aspect of the code is being captured by embedding of the pre-trained models?
- Can embedding from pre-trained model act as an effective input for an automated grading system?

This study is segmented into different sections. The ongoing discussion in Section I provides an overview of the problem statement and outlines the efforts made to potentially address the issue. Section II delves into the literature survey, exploring the approaches taken by other researchers to tackle similar problem statements. Section III outlines the methodology and workflow of the project. Section IV is dedicated to presenting and analyzing the results obtained from the proposed methodology. Finally, Section V serves as the conclusion, summarizing the ultimate findings derived from this project.

II. Literature Survey

Gupta et al. [2] introduce a novel deep learning approach for identifying bugs in student programs using failing tests, eliminating the need for program execution. It employs a unique tree convolutional neural network to predict program

test outcomes and utilizes a state-of-the-art neural prediction attribution technique to identify the responsible lines of code. Evaluated on a substantial dataset of flawed C programs, the technique proves effective in localizing various semantic bugs. The paper's contributions include a new encoding for program ASTs, the integration of prediction attribution in program contexts, and the development of a versatile deep learning technique for bug localization. Kanade et al. [3] introduce CuBERT, a BERT model designed for comprehending code, which underwent training on an extensive dataset comprising 7.4 million Python files sourced from GitHub. CuBERT outperforms Word2Vec, BiLSTM, and Transformer models, as well as cutting-edge counterparts, even with a briefer training period and fewer labeled instances. The model undergoes fine-tuning for multiple classification tasks and a program-repair assignment, showcasing notable accuracy and surpassing other models. Furthermore, the document explores various program-understanding responsibilities, assessing CuBERT's performance against alternative models. This research provides valuable insights into CuBERT's development and its efficacy in comprehending and enhancing source code.

Vasic et al. [4] discus a solution to the VARMISUSE problem using a neural network, which substitutes enumerative search with a pointer network for bug localization and repair. The method attains high accuracy without relying on explicit enumeration and undergoes evaluation using examples from GitHub projects written in C sharp and Python. The paper demonstrates the effectiveness of the model and quantifies the influence of incorrect bug location on repair prediction accuracy. Allamanis et al. [5] introduce an innovative approach to leveraging graph-based deep learning techniques for analyzing program structures. The focus of the study is on creating graphs from source code and adapting the training of Gated Graph Neural Networks (GGNN) to handle large graphs effectively. Specifically, the investigation addresses tasks such as VARNAMING and VARMISUSE, where the network predicts variable names and selects the appropriate variable for a given program location. The paper also delves into the challenges associated with generalizing across diverse source code projects and provides insights into the engineering efforts necessary for implementing GGNNs for sets of large and varied graphs.

Heimer et al. [6] discuss the hurdles and potentials inherent in scrutinizing word vector embeddings, specifically within the realms of linguistics and humanities research. The paper underscores the significance of visualization tools in navigating and comprehending word embeddings, drawing attention to tasks such as examining nearest neighbors, establishing axes for data organization, and evaluating the consistency and disparities among embeddings. Additionally, the paper explores earlier studies that have outlined particular tasks related to analyzing word vector embeddings, illustrating how interactive visualization can streamline these tasks. This study offers valuable insights for the advancement of visualization tools aimed at aiding the examination of word vector embeddings across various research domains. Narmada et al. [7] discuss about the utilization of pre-trained NLP models like T5 and CodeBERT to automate the grading of programming assignments. The paper examines the effectiveness of incorporating transformer embeddings into regression models to replicate the evaluation patterns of human experts. This study offers valuable insights into the use of NLP models for automating the grading of programming assignments, contributing to the progress of automated assessment systems.

Heimerl et al. [8] introduce an innovative approach to comparing two embeddings, with a primary focus on capturing the similarity between objects. The proposed method, embComp, presents overview visualizations based on metrics that measure differences in the local structure around objects. This approach supports various analysis workflows aimed at comprehending similarities and differences within embedding spaces. The paper also explores the utilization of visual analysis techniques such as scatterplots, neighborhood wall view, and distribution comparison metrics. Dundas et al. [9] provide an analysis and assessment of software system quality. The paper accomplishes this by scrutinizing code patterns and employing diverse metrics such as Lines of Code, Execution time, as well as user and developer knowledge. Furthermore, it introduces a classification system for software quality, categorizing it into different levels based on code segments and user interfaces. In essence, this research offers valuable insights for the evaluation and comprehension of software system quality, constituting a substantial contribution to the field of software engineering. Muddaluru et al. [15] propose a method for building an automatic grading system for C programming assignments using regression techniques, deep-learning approaches like convolutional neural networks (CNN) and long short-term memory (LSTM) models; whihc were all applied on the vectors obtained from a transformer model called CodeBERT.

III. METHODOLOGY

A. Dataset

The raw dataset is a combination of solution code along with the question, consisting of 30 unique coding questions, scores for every code, types of errors, correct codes and output, etc. This dataset is converted to embedding using the CodeBERT-unixcoder base nine model which results in 768 features and one target variable. The dataset contains 1176 instances, each representing a solution with its respective question. The project is divided into two parts where different target variables are chosen for each part, one with 'score' attribute and the other with 'Questions' attribute. The score variable, which in the context of this dataset is a score on a range 0-10 (integer values) provided post-evaluation by Subject Matter Experts (SMEs) for every row (code). The dataset contains 30 unique questions which is label encoded (0-29) before adding it as a target variable.

B. Data Pre-processing

The data-cleaning process commences with the imputation of null values with the mean values of the respective feature. The score feature values are changed to integer-type variables from float-type variables. The above experiment is repeated by choosing a different target variable, 'Question' with values in the range 0 to 29.

C. Methodology and Workflow

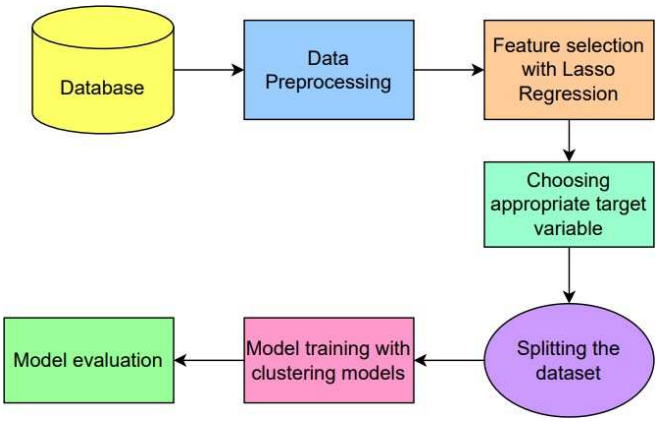

Fig. 1. Architecture of the Proposed Method

Fig.1 shows the detailed workflow of our proposed architecture which is explained below. The project phase is divided into two parts; the entire architecture is applied on two datasets, each time changing the target variable in order to determine which feature results in the best clusters. Initially, the score feature is chosen as the target variable and pre- processing is done. The feature set (X) and a target variable (y) are then extracted from the dataset. Subsequently, the training sets and testing sets are divided up, so they can be used to test the model on unseen data.

Subsequently, the Lasso regression model is trained on the training data, in order to obtain the learned coefficients of the model, effectively performing feature selection. Coefficients represent the weights assigned to each feature in the dataset during the learning process. Randomized Search is then employed for the optimal hyperparameter (α) for the Lasso regression model. The hyperparameter grid is defined using a uniform distribution for α, and the search is conducted with 100 iterations. The Lasso model is then trained using the best-found hyperparameter configuration. Lasso regression is then applied again with the best hyperparameter ($\alpha = 0.01702$) obtained from hyperparameter tuning. After training, the code displays the learned coefficients of the Lasso model, showcasing how the model has adjusted its weights to the features. A new DataFrame is then extracted from the original by selecting the specific features based on the results of the Lasso regression.

The following clustering models are applied to both the subset dataset which only includes the selected features, and the entire dataset:

1) K-means clustering.
2) Agglomerative hierarchical clustering.
3) Density-Based Spatial Clustering.
4) Gaussian Mixture Model.
5) Spectral Clustering.
6) Balanced Iterative Reducing and Clustering using Hierarchies.
7) K-medoids clustering.

When the score feature is chosen as the target variable, all the above-mentioned clustering models are initiated with 11 clusters since the score attribute has scores in the range 0 to 10. Clustering is done on the entire dataset as well as the subset dataset resulting from the feature selection process. The resulting labels for each data point are extracted which indicate to which cluster each data point belongs.

A mapping matrix is generated for each algorithm to assess the performance of the clustering in comparison to the true labels. Subsequently, a cross-tabulation is computed to show the distribution of the target variable 'y' across different clusters. The resulting table provides a tabular representation of how instances from each cluster are distributed across different categories of the target variable. Further, the performances are evaluated using various metrics described in Section IV.

Subsequently, the preprocessing steps are repeated with 'Question' feature as the target. This filtered data was subject to feature selection using Lasso regression, the first step being applying hyperparameter tuning using Randomized Search CV like done for the previous dataset. The Lasso model is then trained using the best-found hyperparameter configuration ($\alpha = 0.015$). A new dataset is created by extracting the selected features.

The following clustering models are applied to both the subset dataset which only includes the selected features, and the entire dataset:

1) K-means clustering.
2) Agglomerative hierarchical clustering.
3) Gaussian Mixture Model.

Clustering is done on the entire data set, including the subset of data identified by the feature selection process. Mapping matrices are generated for each algorithm to evaluate ensemble performance against real benchmarks. A cross-tabulation is then calculated to display the target variable's distribution among the various clusters.

The following metrics are computed to assess clustering quality: Adjusted Rand Index, Normalized Mutual Information, Fowlkes-Mallows Index, Silhouette Score, Calinski-Harabasz Index, and Davies-Bouldin Index. The visualization and metrics collectively offer a comprehensive understanding of how well the algorithm has grouped data points into clusters, considering factors like similarity between true and predicted labels, cluster compactness, and separation. These evaluations aid in quantifying the effectiveness of the clustering approach and its suitability for the given dataset.

IV. Results and Analysis

A. Which machine learning techniques are employed to build the proposed system?

The dataset chosen contains 768 features and one target variable. The features are in the form of code embeddings which are obtained through the CodeBERT-unixcoder base nine model. The main objective of the problem statement is to analyze the clusters made by the machine and compare them to the clusters in the dataset for different target variables and compare the accuracy in each scenario.

For a given target variable, different clustering models are applied such as K-means clustering, BIRCH clustering, Agglomerative clustering, DBSCAN clustering, Gaussian Mixture Model Clustering, Spectral clustering and K-medoids clustering. Each data point's resulting labels are extracted, indicating which cluster each data point is associated with.

B. How to determine the efficiency of a clustering model?

In conclusion, the resulting clusters from the algorithms are evaluated based on the following metrics:

1. Silhouette score
2. Calinski-Harabasz score
3. Davies-Bouldin score
4. Adjusted Rand Index (ARI)
5. Fowlkes-Mallows Index (FMI)
6. Normalized Mutual Information (NMI)

Upon choosing the score variable as the target, the clustering model outputs made by the various clustering algorithms (k-means clustering, BIRCH clustering, Agglomerative clustering, DBSCAN clustering, Gaussian Mixture Model Clustering, Spectral clustering and K-medoids clustering) for the training split of the dataset (X train) are compared to the human graded score target classes for their similarity using the above metrics. A Mapping matrix and a cross table (to depict how many data points of each scoring class belonged to every predicted cluster) are also plotted for each of the models.

Fig.2 shows the Mapping matrix of the K-means Clustering model for the subset dataset when score is chosen as target variable.

All of the above clustering models were implemented with 11 clusters so as to compare with the target 11 clusters in the y variable (scores 0-10). Among the seven clustering models applied, upon analysis of the evaluation metrics, three of them performed comparatively well than the others. These models were, k-means clustering model with a Silhouette score of 0.2973, the agglomerative clustering model with a Silhouette score of 0.2772 and the Gaussian Mixture Model with a Silhouette score of 0.2908.

Table I represents a comparison table for the chosen evaluation metrics resulting from the seven clustering models used for the subset dataset when score is chosen as a target variable.

Table II represents a comparison table for the chosen evaluation metrics resulting from the seven clustering models used for the entire dataset when score is chosen as a target variable.

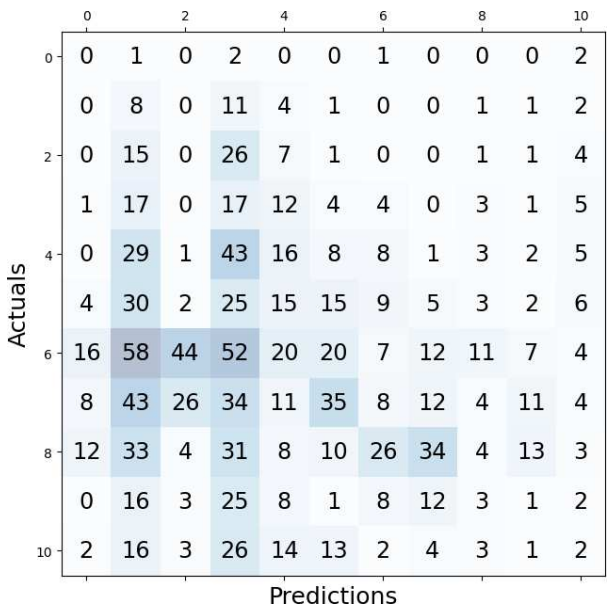

Fig. 2. Mapping matrix of K-means clustering model in the subset dataset with 'score' target variable.

TABLE I
PERFORMANCE ANALYSIS OF CLUSTERING ALGORITHMS WHEN SCORE CHOSEN AS TARGET FOR THE SUBSET DATASET

Clustering Model	Silhouette score	CH score	DB score	ARI	FMI	NMI
K-Means	0.2973	112.1092	1.4681	0.0067	0.1450	0.0705
Agglomerative	0.2772	111.9557	1.5990	0.0360	0.1578	0.0836
DBSCAN	0.0214	23.4761	0.9950	0.0024	0.3571	0.0004
GMM	0.2908	0.0661	1.5011	0.0104	0.1409	0.0661
Spectral	0.0963	48.5951	2.4141	0.0260	0.1795	0.0745
BIRCH	0.2783	106.3074	1.5280	0.0102	0.1439	0.0777
K-Medoids	0.1723	76.9037	1.9741	0.0299	0.1513	0.0637

TABLE II
PERFORMANCE ANALYSIS OF CLUSTERING ALGORITHMS WHEN SCORE CHOSEN AS TARGET FOR THE ORIGINAL DATASET

Clustering Model	Silhouette score	CH score	DB score	ARI	FMI	NMI
K-Means	0.2785	107.5776	1.5680	0.2231	0.1549	0.0710
Agglomerative	0.2876	112.976	1.558	0.0313	0.1527	0.0710
DBSCAN	0.0688	28.503	0.9023	0.0024	0.3571	0.0046
GMM	0.2966	106.1478	1.412	0.0399	0.1752	0.0757
Spectral	0.0719	42.954	2.0618	0.0211	0.200	0.7095
BIRCH	0.2783	106.307	1.5285	0.0102	0.1439	0.0777
K-Medoids	0.1982	74.46	1.944	0.0260	0.1530	0.07005

It is found that there is a high degree of cluster overlapping and this is also reflected in the resulting scores which are not satisfactory when the seven mentioned clustering models are applied to the dataset when score is chosen as the target. The cross tables and the mapping matrices show that the distribution of the clusters amongst different score classes in the actual target variable is not well-defined resulting in overlapping clusters and low scores. This leads to the inference that the machine is not able to cluster the scores in a manner similar to the Subject Matter Experts. This method cannot be applied to build an automated grading system for scoring C codes. Experimentation is further extended by extracting Questions feature from the raw dataset.

Clustering methods are applied to the split dataset after pre-processing and the resulting clusters are compared with the predefined clusters of the target variable using metrics like NMI, FMI, ARI, Silhouette score, Calinski-Harabasz score and Davies-Bouldin score. Mapping matrices and cross tables are plotted for all of the clustering models employed which are, K-means clustering, agglomerative clustering and GMM. The number of clusters are chosen to be 31 for this case since there are 30 unique number of questions in the dataset.

Fig. 3 shows the Mapping matrix of the K-means Clustering model for the subset dataset when Question is chosen as target variable.

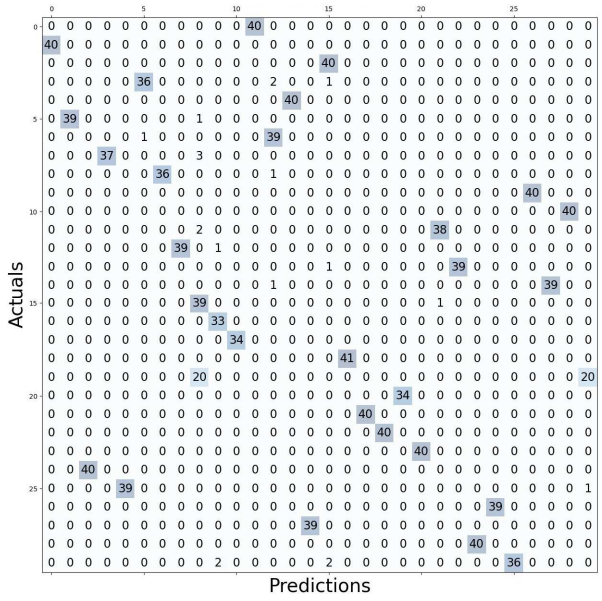

Fig. 3. Mapping matrix of K-means clustering model in the subset dataset with 'Question' target variable.

Fig.4 shows the Mapping matrix of the Agglomerative Clustering model for the subset dataset when Question is chosen as target variable. Fig. 5 shows the Mapping matrix of the Gaussian Mixture Model for the subset dataset when Question is chosen as target variable.

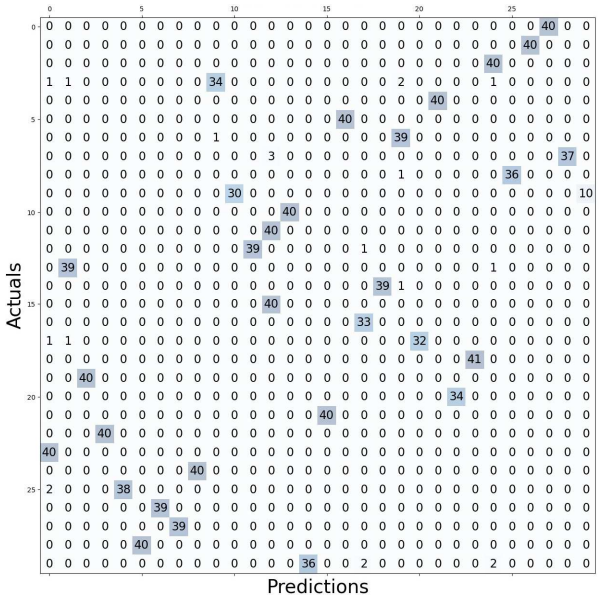

Fig. 4. Mapping matrix of Agglomerative clustering model in the subset dataset with 'Question' target variable.

All of the above clustering models were implemented with 30 clusters so as to compare with the target 30 clusters in the 'y' variable (questions 0-29). It is found that the clusters are not overlapping and are well-defined for the above-mentioned clustering models when Questions is chosen as the target. The cross table is used to further verify the same. The resulting cross table for K-means clustering solidifies the finding that each cluster has data points that mostly belong to a particular class. The cluster distribution amongst different classes is much better as compared to when 'score' was selected as the target variable, proving that the machine is able to cluster the data points based on the question the particular row is dealing with, much better than clustering the rows solely based on their corresponding scores, as obtained by clustering models.

C. Which is the best model for the system?

Table IV represents a comparison table for the chosen evaluation metrics resulting from the three clustering models used for the subset dataset when Question is chosen as a target variable.

TABLE III
PERFORMANCE ANALYSIS OF CLUSTERING ALGORITHMS WHEN QUESTIONS FEATURE CHOSEN AS TARGET FOR THE SUBSET DATASET

Clustering Model	Silhouette score	CH score	DB score	ARI	FMI	NMI
K-Means	0.5615	249.64	0.8219	0.9341	0.9364	0.9694
Agglomerative	0.5772	249.65	0.7762	0.9177	0.9210	0.9666
GMM	0.5516	242.03	0.7707	0.8886	0.8943	0.9643

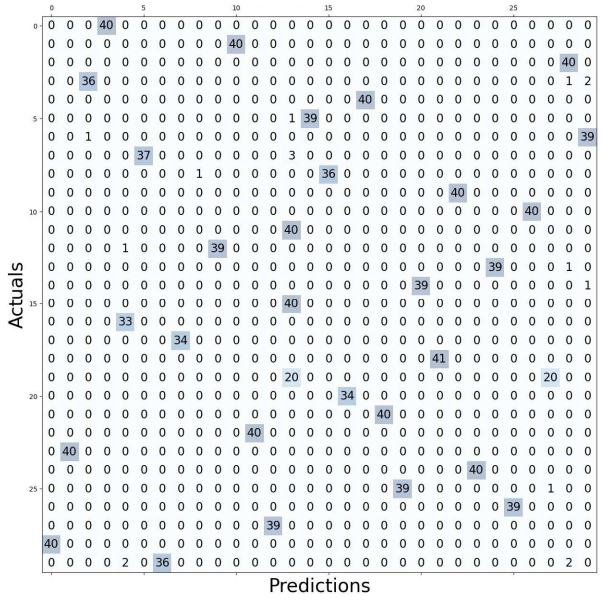

Fig. 5. Mapping matrix of Gaussian components in the subset dataset with 'Question' target variable.

Table IV represents a comparison table for the chosen evaluation metrics resulting from the three clustering models used for the entire dataset when Question is chosen as a target variable.

TABLE IV
PERFORMANCE ANALYSIS OF CLUSTERING ALGORITHMS WHEN QUESTIONS FEATURE CHOSEN AS TARGET FOR THE ENTIRE DATASET

Clustering Model	Silhouette score	CH score	DB score	ARI	FMI	NMI
K-Means	0.5902	274.74	0.7392	0.9217	0.9248	0.9700
Agglomerative	0.6106	287.28	0.7376	0.9103	0.9138	0.9639
GMM	0.6127	295.21	0.6871	0.9217	0.9248	0.9700

From amongst the above three models employed, the analysis of various scores proves that, K-means clustering provides the most well defined clusters when Questions variable is chosen as target. Another important observation made is that depending on the clustering model used, the dataset composed of only the selected features may or may not yield better results when compared to the results of the clustering model applied to the entire dataset. A possible explanation for this, is the fact that a particular model might not be sensitive to the selected features in comparison to another model, i.e., it may not utilize the additional information provided by certain features, making their inclusion or exclusion less impactful.

D. Can embedding from pre-trained model be used as an input for automated grading systems?

In an attempt to map the scores with the questions to gain information from the cross tabulations, K-means clustering was performed with number of clusters as 11 on the score feature and the resulting clusters were compared to the questions feature with 30 unique questions. The obtained cross table gives information about how the scores are distributed amongst different questions. After this, a new variable 'Label' was added to the dataset, which consisted of mapped values of 330 different classes obtained by different combinations of questions and scores in the form of (Questions, score). Each of the combinations was assigned an integer class label as a combination of question label + score and this was taken as the target variable for further clustering by K-means with 330 clusters. The above-mentioned evaluation metrics were applied and the results obtained give a silhouette score of 0.31, Calinski Harabasz Score of 185.4524, Davies Bouldin Score pf 0.6673, ARI Score of 0.0, NMI Score of 0.0 and a Fowlkes Mallow Index of 0.0772. It can be inferred from the above metrics that the clusters overlap like in the first case when clustering was done at the score level. The clusters are not very compact either.

TABLE V
QUESTIONS 11 AND 15 WHICH ARE EVALUATED FOR THEIR SIMILARITIES

Question No.	Question Name
11	Write a C program to merge two arrays
15	Merge two sorted arrays in C

Table V shows Questions 11 and 15 along with the respective Question names. Further, based on the results of the mapping matrix - question 11 and question 15 were picked to understand why they were grouped together in all clustering algorithms. Table V shows the questions present in 11 and 15 respectively. It can be seen that both the questions focus on merging two arrays with one particularly focusing on 'sorted' arrays . To see how similar the solutions for both the questions were, cosine similarity were applied and it was found that they were similar by 69 % . Thus, it can be said that embeddings are able to capture the essence of the concept being asked more accurately but the embedding is not able to distinguish and classify the correct code solution and the code solution with errors and the type of errors. Clustering at a score level is overlapping and the evaluation metrics do not give any optimal values thereby proving the embeddings are not able to capture the type of error and are rather just clustering the codes based on the similarity of the text. The logic behind the solution is being ignored and only the concept behind the question being asked is being taken into consideration in the created embeddings. Thus, on clustering, it gives well compact clusters for Questions as the target variable.

V. CONCLUSION

The proposed study focuses on the application of clustering techniques in order to analyze patterns and anomalies in the

embedded versions of c program solutions to various problem statements via the grading technique implemented by SMEs. The application of various clustering models using different target variables each time (score and Questions), has led to the inference that, the given CodeBERT embedding can cluster efficiently at a questions level but fails to do so at the score level. The reason behind this could be that the model ignores logic and clusters similar texts together. From this research, it can be concluded that, for this particular dataset, the embedding from CodeBERT do not capture the degrees or the types of errors present in the code, but it rather looks at it from a more graphical level and hence maps the questions accurately. Further, these embeddings can be analyzed based on the common clusters formed based on scores. Also, embeddings from fine-tuned models can be explored.

ACKNOWLEDGEMENT

Amrita Vishwa Vidyapeetham provided the necessary infrastructure and support for the conduct of this research activity, as well as for the production of the publication, for which the authors are grateful. Additionally, the authors would like to acknowledge the usage of open source AI tools like ChatGPT.

REFERENCES

[1] A. Kanade, P. Maniatis, G. Balakrishnan and K. Shi, "Learning and evaluating contextual embedding of source code", In International conference on machine learning, 2020, pp. 5110-5121. PMLR. arXiv:2001.00059v3

[2] R. Gupta, A. Kanade and S. Shevade, "Deep learning for bug-localization in student programs", arXiv:1905.12454, 28 May, 2019.

[3] J. B. Dundas, "Understanding Code Patterns Analysis, Interpretation and Measurement", In 2008 International Conference on Computer and Electrical Engineering, 2008, pp.150-154, IEEE. doi: 10.1109/ICCEE.2008.174

[4] M. Vasic, A. Kanade, P. Maniatis, D. Bieber, and R. Singh, "Neural program repair by jointly learning to localize and repair", arXiv preprint, 2019, arXiv:1904.01720.

[5] M. Allamanis, M. Brockschmidt and M. Khademi, "Learning to represent programs with graphs", arXiv preprint, 2017, arXiv:1711.00740.

[6] F. Heimerl and M. Gleicher, "Interactive analysis of word vector embeddings", In Computer Graphics Forum, 2018 (Vol. 37, No. 3, pp. 253-265). doi:10.1111/cgf.13417

[7] N. Nakka and P. B. Pati, "Autograding of Programming Skills", In 2023 IEEE 8th International Conference for Convergence in Technology (I2CT), 2023, pp. 1-6, IEEE. doi: 10.1109/I2CT57861.2023.10126211

[8] F. Heimerl, C. Kralj, T. Mo¨ller and M. Gleicher, "embComp: Visual Interactive Comparison of Vector Embeddings", in IEEE Transactions on Visualization and Computer Graphics, vol. 28, no. 8, pp. 2953-2969, 2022, doi: 10.1109/TVCG.2020.3045918

[9] J. B. Dundas, "Understanding Code Patterns Analysis, Interpretation and Measurement", 2008 International Conference on Computer and Electrical Engineering, Phuket, Thailand, 2008, pp. 150-154, doi: 10.1109/ICCEE.2008.174

[10] Y. Shi, K. Shah, W. Wang, S. Marwan, P. Penmetsa and T. Price, "Toward semi-automatic misconception discovery using code embeddings", In LAK21: 11th International Learning Analytics and Knowledge Conference, Association for Computing Machinery, New York, NY, USA, 2021, pp. 606-612, arXiv:2103.04448v1

[11] G. Haldeman, A. Tjang, M. B. Vroman, S. Bartos, J. Shah, D. Yucht and T. D. Nguyen, "Providing Meaningful Feedback for Autograding of Programming Assignments", In Proceedings of the 49th ACM Technical Symposium on Computer Science Education (SIGCSE '18). Association for Computing Machinery, New York, NY, USA, 2018, pp. 278–283, doi: 10.1145/3159450.3159502

[12] M. S. Vineeth, K. RamKarthik, M. S. Phaneendra Reddy, N. Surya and L. R. Deepthi "Comparative analysis of graph clustering algorithms for detecting communities in social networks", In Ambient Communications and Computer Systems, Advances in Intelligent Systems and Computing, vol 1097. Springer, Singapore.: RACCCS, 2019, pp. 15-24, doi: 10.1007/978-981-15-1518-7_2

[13] A. Abraham, P. Dutta, J. K. Mandal, A. Bhattacharya, and S. Dutta. "Emerging Technologies in Data Mining and Information Security. Advances in Intelligent Systems and Computing", Proceedings of IEMIS-2018 (2018). doi: 0.1007/978-981-15-9774-9

[14] R. V. Muddaluru, S. R. Thoguluva, S. Prabha, P. B. Pati and R. M. Balakrishnan, "Auto-grading C programming assignments with CodeBERT and Random Forest Regressor", 2023 14th International Conference on Computing Communication and Networking Technologies (ICCCNT), Delhi, India, 2023, pp. 1-6, doi: 10.1109/ICCCNT56998.2023.10308341.

[15] P. Manasa, P. Ananth, P. Natarajan, K. S. Sundaram, E. R. Rajkumar, K. S. Ravichandran, A. H Gandomi, "An Analysis of Causative Factors for Road Accidents using PAM and Hierarchical Clustering Techniques", Engineering Reports, accepted, 2023, doi: 10.1002/eng2.12793

[16] K. Rajanbabu, I. K. Veetil, V. Sowmya, E. A. Gopalakrishnan, and K. P Soman. "Ensemble of Deep Transfer Learning Models for Parkinson's Disease Classification". In Soft Computing and Signal Processing: Proceedings of 3rd ICSCSP 2020, Volume 2, pp. 135-143. Springer Singapore. doi: 10.1007/978-981-16-1249-7_14

[17] L. R. Chandran, K. Ilango, M. G. Nair, A. A. Kumar, and 2022, "Multilabel external fault classification of induction motor using machine learning models". In 2022 Third International Conference on Intelligent Computing Instrumentation and Control Technologies (ICICICT), pp. 559-564, IEEE. doi: 10.1109/ICICICT54557.2022.9917882