Optimizing Vector Representations in LLMs:A Break-and-Append Approach for Code Inputs

G Mokshith Teja, M Himesh Reddy, Y V Abhiram, MS.Nakka Narmada, Dr.Peeta Basa Pati
Department of Computer Science and Engineering
Amrita School of Computing, Bengaluru
Amrita Vishwa Vidyapeetham, India
bl.en.u4aie23041@bl.students.amrita.edu, bl.en.u4aie23059@bl.students.amrita.edu,
bl.en.u4aie23065@bl.students.amrita.edu, bp_peeta@blr.amrita.edu

***Abstract*—Large Language Model NLP technologies exhibit excellent performance in processing human language with advanced evaluation metrics. Vector representation optimization is a key issue that impacts NLP systems despite optimal performance by such technologies. The main processing issue in code programming arises during optimization when data types need to be converted. The project solves this issue with a break-and-append strategy that enables learning vector representations with high accuracy in semantic encoding and syntactic structuring. The strategy divides source code elements into meaningful parts that improves the processing speed and enables computer systems to reconstruct code automatically. The system uses programming structures with dependency mechanisms to construct strong contextual understanding which enhances pattern recognition capability as well as operational performance and accuracy.**

This methodology enables programmers to attain definition generation as well as error detection through its programming function support. Vectorization techniques enable model performance optimization through its capacity to harness computational efficiency. Quantitative measurement methods demonstrate improved performance results through its representation optimization in the research findings. The research develops a model that integrates natural language and programming languages to realize improved domain interoperability between processing systems. NLP-based system code processing performance is optimized through this method yielding positive results for next-generation LLM development as well as improving processing capacity for complex data structures.

***Index Terms*—Vector Representation, Semantic Encoding, Optimization Strategie**

I. Introduction

In the dynamic field of NLP, LLMs have shown great performance in understanding and generating human-like text. However, optimization of vector representations remains a challenge in structured data inputs, such as programming code. This project introduces an advanced method to achieve accurate vector representations through the integration of break-and-append mechanisms within LLM architectures. This method is one that seeks to enrich the semantic encoding of the code inputs for acquiring effective syntactic structures together with their attendant dependencies present within the context. The process reduces code into significant pieces or bits (breaks) and intelligently reconstructs this code, bringing about improved aptitude on its part for managing complex patterns and hence raising better performance for various tasks that incorporate code generation or summarization through error detection also. Further, the paper experimentally verifies through quantitative evaluation whether different techniques in vectorization help improve model performance and are computationally efficient for practical applications. Overall, these insights are conducive to the building of more powerful LLMs that can operate across a broad set of representations—from natural language, to code-to bridge gaps across both languages of processing.

II. Literature Survey

[1] This paper presents ChunkBERT as an efficient fine-tuning strategy for BERT models in the context of long-text classification. The novelty involves chunking of long inputs into smaller, more manageable pieces that are then independently processed using CNN layers. BERT's limitation to 512 tokens is then circumvented while significantly reducing memory usage (to 6.25 percent of the original). On a long-text benchmark, the evaluation shows that ChunkBERT is more resource-efficient than previous long-sequence models but maintains competitive performance across a wide range of classification tasks.[2]Late Chunking is a novel chunking technique that, instead of chunking the text, encodes the whole document using a long-context embedding model and only chunks the text afterwards. In this way, the chunk representations do not lose the global context like traditional chunking methods do by breaking the texts into smaller parts before embedding. This approach brings significant improvement in retrieval performance for neural IR and RAG tasks by capturing richer semantic dependencies. Besides, the authors propose a fine-tuning method to increase the effectiveness of retrieval. Experiments show better retrieval accuracy compared to naive chunking.[3]This paper critically reviews the effectiveness of semantic chunking for retrieval-augmented generation systems. Popular though semantic chunking methods-which split a document into meaningful, coherent subgroups-have proved to be; benefits are erratic and do not appear to balance additional computational expense. Results from our experiments showed that fixed-size chunking - simply splitting

the text into fixed-size chunks-pared or did even better on most retrieval and answer generation tasks. Findings rejected the assumption about the superiority of semantic chunking and proposed more efficient and adaptive chunking strategies. [4]The paper introduces "Late Chunking" as a novel technique aiming at enhancing text chunk embeddings in dense vector-based retrieval systems by mitigating the loss of contextual information observed with the traditional chunking methods. Instead of splitting text before embedding, late chunking first processes the whole document using long-context embedding models and carries out chunking just before the mean pooling step. This method, on the other hand, lets each chunk capture global contextual information, contributing thus to better retrieval performance without any additional training required. Experimental results on BeIR benchmark datasets show that late chunking consistently outperforms naive chunking methods, especially for longer documents when context preservation becomes critical.[5]A new detection method for MGC based on CodeT5+ with the statistical analysis of Log-Likelihood and Rank and Entropy metrics appears in the MAGECODE paper. This technique defines how to distinguish between AI-generated code and human-written code to address the common problem of academic integrity that occur in educational settings. This research demonstrates Log-Likelihood is the top choice for detecting MLC in Python and C++ programming code while Entropy achieves the best performance in Java. This research introduces over 45,000 machine-generated code samples from GPT-4, Gemini and Code-bison-32k which serve as benchmark data for evaluating detection systems. The reliability of MAGECODE includes 98.46 percent detection accuracy along with a productive false positive ratio under 1percent to help protect educational and professional sectors.[6]The CoLLEGe paper develops a meta-learning structure which enables large language models to acquire few-shot concept knowledge. Traditional in-context learning experiences some distractibility but CoLLEGe develops adaptable embeddings with limited examples to help people learn words and define concepts and reason verbally. CoLLEGe demonstrates its performance in dealing with GRE-style fill-in-the-blank problems alongside internet slang interpretation and definition generation. The performance of CoLLEGe surpasses both HiCE and Token Tuning when LLMs must acquire new concepts without explicit training in zero-shot conditions. The trained embeddings become more effective through example buffers together with negative sampling and knowledge distillation methods without requiring specific training for each task. The functionality strengthens LLMs to adapt better since they gain enhanced abilities to learn new concepts through dynamic incorporation of information.[7]This section analyzes the CoLLEGe paper which implements a meta-learning framework for teaching concepts to LLMs through few-shot examples. CoLLEGe employs flexible embedding generation with limited examples to overcome the distraction of traditional context-learning approaches and enhance word learning and definition interpretation and verbal communication. The evaluation process uses demanding real-world challenges which include GRE-style fill-in-the-blank tests together with internet slang recognition and definition output tasks. CoLLEGe achieves superior performance than HiCE and Token Tuning when evaluated in zero-shot learning conditions which enable LLMs to easily adopt new concepts. The system combines example buffers alongside negative sampling together with knowledge distillation to enhance embeddings through training that does not need extra task-specific training steps. The research builds LLM adaptability to dynamically perceive and understand new concepts through better integration of dynamic information processing.[8]The document gives a computational framework for explaining the processing of cognitive systems on environmental contingencies. The research group posits that contextual learning can arise from changes in shift-based focus between stimulus triggering. Via its self-learning process, the model associates stimuli with specific loci and resorts to novel detection against predicted stimuli for updating the context representations of the model. The approach itself is verified by simulations of conditioning and habituation which shows the impact of contextual learning on memory functions. The article argues with regard to two areas of the brain called hippocampus and prefrontal cortex since these components play an essential role in context-based learning. Research has shown how sequential stimulus integration creates contextual representations, which aids artificial intelligence development and also progress in cognitive modeling.[9]The research analyzes Deep Neural Network models intended for NLU Sequence Chunking operations which consist of text chunking tasks along with semantic slot filling procedures. According to the authors the traditional sequence labeling method using IOB demonstrates restricted application as it does not treat chunks as fully independent units. The researchers created three neural network structures and included one Bi-LSTM model together with encoder-decoder and pointer network models. The pointer-based model achieves state-of-the-art performance by surpassing the other proposed models in segmenting and labeling chunks according to test results. The study demonstrates that explicit segmentation plays a crucial part in chunking together with the proposal of deep learning models as effective tools for organized linguistic data processing.[10]The analysis investigates NLP models BERT and CuBERT for their operation as software code prediction systems. The paper verifies its conclusions by conducting an evaluation between new models and established metric-based regression systems. CuBERT serves as a better code prediction model because it received training with code-related datasets. With domain-specific training the authors discovered that CuBERT produced superior outcomes during fine-tuning procedures. Embeddings derived from NLP serve applications in software development to develop efficient code analysis systems and change prediction models. Research helps document software development patterns by providing data-based tools to help software activities.

III. Methodology

A.WORKFLOW DIAGRAM

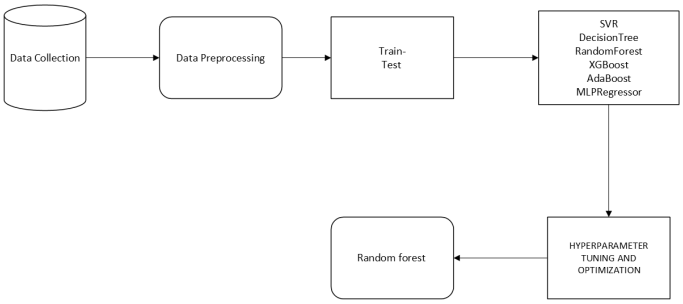

Fig. 1. WorkFlow of Models

B.FEATURE ANALYSIS AND DATA PREPROCESSING Machine learning uses feature-based analysis of the information within datasets to achieve data integrity quality objectives. Phase one begins with proper imputation methods and conversion processes for numeric attributes prior to standardizing necessary variables. Relationship evaluations via matrix analysis facilitate the selection of key features that eliminate unnecessary features with very high correlations among them. The model performs improved performance via this approach because it integrates dimension reduction methods with overfitting prevention methods.

The integration of scatter plots and histograms and correlation heatmaps allows users to view feature relationships and also notice distribution patterns. Statistical techniques of Interquartile Range (IQR) and Z-score identify harmful outliers in data that would adversely affect model performance and eliminate them completely. A normalization step allows equal feature scaling since some algorithms show sensitivity towards features of varying sizes and scales. By applying 80-20 division method the preprocessed data are divided into training subsets in addition to testing subsets with which the model gets appropriate training for evaluation by held-out evaluation data. The target variable is predicted using multiple regression models in which SVR, Decision Tree, Random Forest and XGBoost, AdaBoost and MLP Regressor are used.

Performance evaluation of the models covered RMSE and MAE evaluation measures as well as R-squared (R^2). Several accuracy and reliability performance metrics assist in pinpointing reliable prediction performance and expand application limits of modeled predictions. Model selection continues to compare findings with each other for determining the optimal optimization solution.

C. MODEL TRAINING AND EVALUATION
Following feature analysis halt the training of Decision Tree classifier began. Data for testing purposes at 20% and the amount of data for training 80% was provided to the model for the general predictive examination. Design for the framework utilized a depth limitation constraint in a bid to meet training stability with decision models created that were easily comprehensible.The Plot_tree capability made Release Tree display in showing us the feature divisions a model makes choices based on. The visual layout illustrates how recursive classification rules develop their decision process. Visual was required to review the classified area. The predictions from the model output indicate the sets of data that each of the mesh grid cells classifies by. A collection of standard measures assessed the Decision Tree classifier by combining measures of accuracy with precision and recall standards and F1-score standards. The metric system assists the classifier in decomposing target classes through error reduction across groups. The use of these techniques improves model reliability by utilizing practices following machine learning best standards in feature selection and transformation and data classification. A multistep process offers a comprehensive evaluation system through steps that begin with entropy calculations followed by bins creation and scaling implementation and tree construction before establishing visible decision boundaries for robust classification results.

D.HYPERPARAMETER TUNING AND OPTIMIZATION RandomizedSearchCV is the utility to perform hyperparameter tuning for better model performance. During the optimization process the best set of hyperparameters for the Random Forest model are discovered by it which includes the number of estimators in addition to the maximum depth in combination with the minimum samples split and minimum samples leaf and max features. Subsequent to the retraining of the model on chosen optimum parameters the evaluation again occurs through test set evaluations in order to exhibit improved performance in comparison to baseline models.

The model applies cross-validation measures to avoid overfitting and ensure uniform performance across the different subsets of data. Feature importance analysis identifies significant variables for prediction that allow the model to be improved through focusing on significant factors for future growth. The integration of weak learners in ensemble learning methods such as bagging and boosting more stabilizes model reliability to improve prediction accuracy levels.

IV. Result Analysis and Discussion

Preprocessing methods in combination with feature selection techniques resulted in improved results in regression model predictive power. Correlation matrix analysis was used to identify correlated features and thus eliminate unnecessary attributes which improved prediction models as well as their interpretations. StandardScaler standardization allowed all features to have the same scales so that models would not lean towards attributes with greater magnitudes for prediction.

Ensemble regression model methods Random Forest and XGBoost performed better than single estimators Decision Tree and SVR based on test data evaluation results and training results. Random Forest demonstrated the best generalization capability since it achieved optimal bias and variance levels. Model precision enhanced by the hyperparameter tuning process was responsible for selecting the best parameter sets for prediction results.

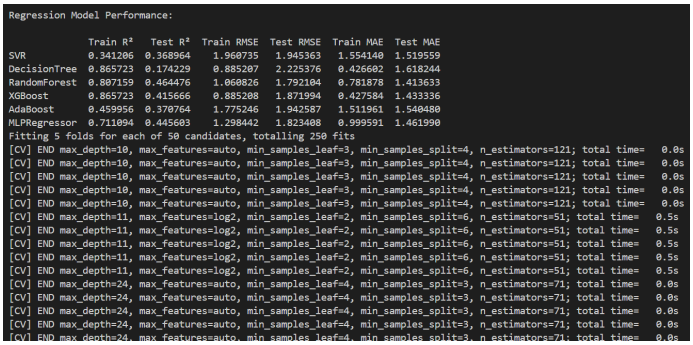

Fig. 2. Train-Test Score

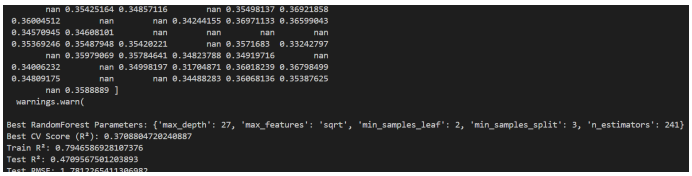

Fig. 3. Score

Evaluation findings validated Random Forest as the better method since it yielded the greatest R^2 score with the least error rates by using metrics RMSE and MAE. Findings of the residual analysis substantiated the model's reliability to predict through the normal distribution of errors.

This research revealed that effective feature selection methods in combination with preprocessing techniques and ensemble learning methods propel the performance level of regression models to greater heights. Accurate and interpretable predictive machine learning models necessitate systematic hyperparameter tuning that goes hand in hand with stringent evaluation metrics.

REFERENCES

[1] Günther, Michael, Isabelle Mohr, Daniel James Williams, Bo Wang, and Han Xiao. "Late chunking: contextual chunk embeddings using long-context embedding models." arXiv preprint arXiv:2409.04701 (2024).

[2] Jaiswal, Aman, and Evangelos Milios. "Breaking the Token Barrier: Chunking and Convolution for Efficient Long Text Classification with BERT." arXiv preprint arXiv:2310.20558 (2023).

[3] Qu, Renyi, Ruixuan Tu, and Forrest Bao. "Is Semantic Chunking Worth the Computational Cost?." arXiv preprint arXiv:2410.13070 (2024).

[4] Kaibe, Yuto, Hiroyuki Okamura, and Tadashi Dohi. "Towards Predicting Source Code Changes Based on Natural Language Processing Models: An Empirical Evaluation." 2023 IEEE 34th International Symposium on Software Reliability Engineering Workshops (ISSREW). IEEE, 2023.

[5] Pham, Hung, et al. "MAGECODE: Machine-Generated Code Detection Method Using Large Language Models." IEEE Access (2024).

[6] Zhai, Feifei, et al. "Neural models for sequence chunking." Proceedings of the AAAI conference on artificial intelligence. Vol. 31. No. 1. 2017.

[7] Schmidt, Craig W., et al. "Tokenization Is More Than Compression." arXiv preprint arXiv:2402.18376 (2024).

[8] Balkenius, Christian, and Jan Morén. "A computational model of context processing." 6th international conference on the simulation of adaptive behaviour. 2000.

[9] Teehan, Ryan, Brenden Lake, and Mengye Ren. "CoLLEGe: Concept Embedding Generation for Large Language Models." arXiv preprint arXiv:2403.15362 (2024).