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

G Mokshith Teja, M Hiimesh 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.u4cse23041@bl.students.amrita.edu, bl.en.u4cse2059@bl.students.amrita.edu,
bl.en.u4cse23065@bl.students.amrita.edu, bl.en.id.cse22002@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

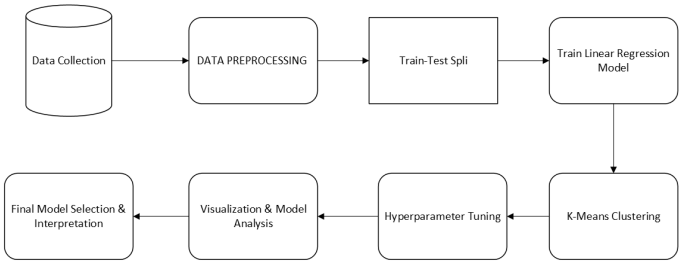

Fig. 1. WorkFlow of given Work

B. FEATURE ANALYSIS AND DISTANCE MEASUREMENT

The use of machine learning methods significantly relies on feature evaluation that assists in determining dataset attributes. The analysis process necessitates examination of dataset structure along with detection of missing records and omitted outlier points in addition to inspection of variable distribution forms. You can comprehend feature relationships by employing quantitative evaluations in addition to graphical displays with histograms and correlation matrices. The model performance enhances via feature selection as researchers retain only necessary variables based on their applicability.

Calculations of distances are a necessary variable when using machine learning models specifically to cluster algorithms. Points of data are examined using Minkowski distance since this framework encompasses both Euclidean and Manhattan distance methods as part of its similarity measure. The model considers distances among members within groups as well as between various groups in order to check proper classification distribution. K-Means clustering utilizes such calculations for accomplishing grouping purposes as it discriminates among homogeneous instances from heterogeneous ones. Through the process of implementing distance metrics the model gets improved features in order to identify data patterns that further enhance both classification as well as clustering results.

C. MODEL TRAINING AND EVALUATION

After feature analysis and preprocessing, the dataset undergoes training and testing to construct an efficient model performance analysis. Linear regression can be applied for predictive modeling through the analysis of the relationship between independent variables and the target variable. The prediction accuracy is gauged through four performance measures of evaluation which are Mean Squared Error (MSE), Root Mean Squared Error (RMSE) and Mean Absolute Percentage Error (MAPE) and R-squared (R^2).

K-Means clustering operates to identify clusters of similar data points on the basis of their relation between the features. The clusters are obtained using the Elbow Method and the model is evaluated using three different metrics like Silhouette Score and Calinski-Harabasz Score and Davies-Bouldin Index. The evaluation metrics ensure that the clusters obtained have well-defined boundaries among them.

IV. RESULT ANALYSIS AND DISCUSSION

Different pairings of the clustering algorithm with regression techniques decide the final accuracy levels and trends in R^2-score. The feature selection of linear regression achieved impressive accuracy in the prediction of final marks that led to perfect classification of the target variable. Employing k-Means clustering led to erratic results in splitting clusters since it introduced inconsistency in silhouette scores and Davies-Bouldin indices. By raising the value of k parameter from 2 to 20 led to lower intra-cluster dispersion that enhanced the clustering outcome. According to the Elbow Method the researchers found a value of k which achieved the right balance between good clustering and adequate separation between classes. The values of k decide the similarity of the data points into clusters where lower values lead to high intraclass cohesion but the increase in the values leads to confusing partitions among the points.

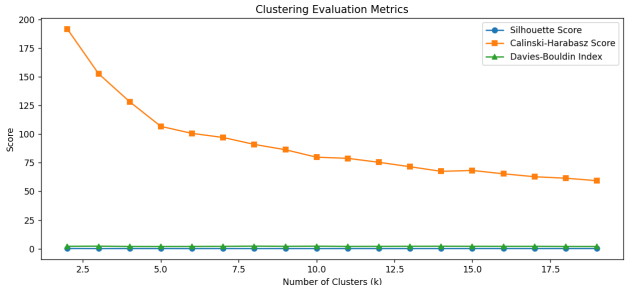

Fig. 2. Clustering Evaluation Metrics

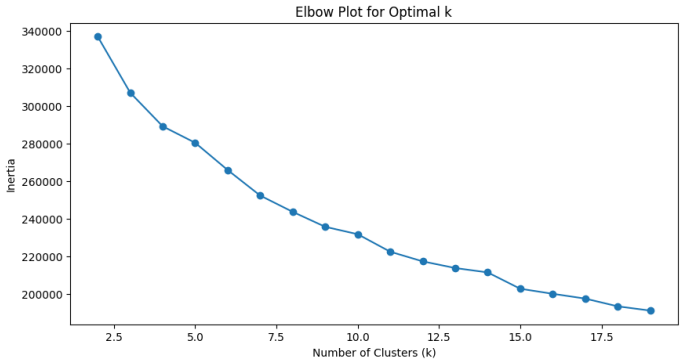

Fig. 3. Elbow Plot for Optimal K

A grid search optimization process was done that resulted in the determination of the optimal value of k. The chosen cluster value k=15 yielded optimal results by offering the proper match between data points of the same cluster and those which were in between clusters. The increase in k beyond its optimal value did not raise the levels of accuracy but introduced complexity that made the choice of the optimal number of clusters as justified. A performance evaluation through silhouette score and Calinski-Harabasz score with Davies-Bouldin index as the metrics showed the effectiveness of the chosen k value determination. The research emphasizes

the significance of the right choice of k value to yield meaningful classification results.

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).