PARAPHRASE IDENTIFICATION AND COMPARATIVE ANALYSIS WITH FINE TUNED AND PROMPT BASED LLM MODELS

RAHUL NANDI

Final Thesis Report

MARCH 2025

TABLE OF CONTENTS

ACKNOWLEDGEMENTS

I would like to express my heartfelt gratitude to Dr. Dattatray for his invaluable support, expert guidance, and encouragement during the crucial stages of this research. His insights and mentorship have been instrumental in shaping the direction and depth of my work.

I extend my sincere appreciation to Liverpool John Moores University (LJMU) and Dr. Manoj for their academic guidance and for sharing valuable information that significantly contributed to the quality of this thesis. Their support has been essential in broadening my understanding of the subject.

I would also like to acknowledge UpGrad for providing a robust learning platform that enabled me to pursue a Master's degree from an esteemed international institution. Their well-structured curriculum, access to industry-relevant resources, and seamless collaboration with LJMU have been vital in facilitating my academic and professional growth.

Additionally, I would like to acknowledge my own dedication, perseverance, and hard work in completing this research. Undertaking this thesis independently was both a challenge and an opportunity for growth. The countless hours of analysis, learning, and problem-solving have strengthened my expertise and deepened my passion for research.

Finally, I am grateful to my family and friends for their encouragement and patience throughout this journey. Their unwavering support kept me motivated even in the most demanding moments.

Thank you all for being a part of this journey.
Rahul Nandi

DEDICATION

To my beloved parents, the guiding stars of my life your endless sacrifices, unwavering belief, and unconditional love have been the foundation of my journey. Every milestone I achieve is a reflection of your relentless support and encouragement.

&

To my loving wife, my pillar of strength and my greatest companion your patience, understanding, and unwavering faith in me have been the light that carried me through this journey. Your love is my greatest motivation, and this accomplishment would not have been possible without you.

With deepest gratitude and love, I dedicate this work to you—the ones who have stood by me through every challenge and triumph.

ABSTRACT

Paraphrase Identification is a critical task in Natural Language Processing (NLP) that determines whether two sentences convey the same meaning. This research investigates the effectiveness of prompt-based, fine-tuned encoder-based, decoder-based, and encoder-decoder-based Large Language Models (LLMs) for paraphrase identification. Using the Microsoft Research Paraphrase Corpus (MRPC), we evaluate models across four approaches: prompt-based (GPT-4o-mini, GPT-3.5-Turbo, Llama-3.2-1B, Llama-3.2-3B, Mistral-7B), fine-tuned decoder-based (Llama-3.2-1B, Llama-3.2-3B, Mistral-7B), fine-tuned encoder-based (ModernBERT), and fine-tuned encoder-decoder-based (T5). The study employs fine-tuning strategies such as QLoRA (Quantized Low-Rank Adaptation) & LoRA (Low-Rank Adaptation) to optimize model performance while minimizing computational costs. Results show that fine-tuned models consistently outperform prompt-based approaches in terms of accuracy. Among fine-tuned models, encoder-based models achieve the highest accuracy (88%), followed closely by encoder-decoder-based models (86%) and decoder-based models (84%). While decoder-based models perform well, they demand significantly higher computational resources, making them less efficient for large-scale applications. In contrast, encoder-based models achieve comparable accuracy with reduced computational overhead, making them a more practical alternative. Prompt-based models, while computationally efficient, achieve the lowest accuracy (76%) and are highly sensitive to prompt design, leading to variability in responses. This study contributes to NLP by providing insights into the trade-offs between these approaches and recommending efficient fine-tuning techniques for resource optimization. These findings guide model selection based on accuracy, computational constraints, and deployment feasibility in real-world applications.

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

AUC	Area Under Curve
BERT	Bidirectional Encoder Representations from Transformers
BPE	Byte-Pair Encoding
ELMo	Embedding from Language Models
GELU	Gaussian Error Linear Units
GLUE	General Language Understanding Evaluation
GPT	Generative Pre-Trained Transformers
GQA	Group Query Attention
HF	Hugging Face
ICL	In-context learning
KV Cache	Key-Value Cache
LDA	Latent Dirichlet Allocation
LLM	Large Language Models
Llama	Large Language Model Meta AI
LoRA	Low-Rank Adaptation
MLM	Masked Language Modeling
MoE	Mixture of Experts
MRPC	Microsoft Research Paraphrase Corpus
MT-DNN	Multi-Task Deep Neural Network
NLM	Neural Language Model
NLP	Natural Language Processing
NSP	Next Sentence Prediction
PAWS	Paraphrase Adversaries from Word Scrambling
PEFT	Parameter Efficient Fine-Tuning
PI	Paraphrase Identification
QLoRA	Quantized Low-Rank Adaptation
QQP	Quora Question Pairs
ReLU	Rectified Linear Unit
ROC	Receiver Operating Characteristics

RoPE	Rotary Positional Encoding
RMS Norm	Root Mean Square Layer Normalization
SICK	Sentences Involving Compositional Knowledge
SwiGLU	Swish-Gated Linear Units
T5	Text-To-Text Transfer Transformer
TF-IDF	Term Frequency Inverse Document Frequency

CHAPTER 1

INTRODUCTION

1.1 Background of the Study

Text similarity is one of the ways to identify whether two pairs of sentences or sequences are identical or not. If identical, then it is called as paraphrase, otherwise non-paraphrase. Paraphrase Identification is beneficial for several NLP tasks like plagiarism detection, question-answering performance improvement, text summarization, and machine translation etc., I am used to figure out whether two text sequences or documents convey the same meaning or not. In the research field, Paraphrase Identification can be used to evaluate plagiarism detection. In the education field, it is used to determine whether the answer is semantically equivalent to a reference answer or not. In summarization, it is used to eliminate redundant information.

To identify paraphrases or text similarity effectively, two key approaches are commonly used-lexical similarity and semantic similarity.

Lexical similarity: The lexical similarity focuses on how similar two text sequences are in terms of their surface-level features, such as word overlap, structure, or syntax. For example, the phrases are “the cat ate the mouse” and “the mouse ate the cat food”. By just looking at the words? On the surface, if you consider only word level similarity, these two phrases appear very similar as 3 of the 4 unique words are an exact overlap.

Semantic Similarity: The semantic similarity focuses on how similar two sequences are in terms of their meaning rather than surface level. Methods of calculating semantic similarity are cosine similarity, Jaccard similarity, word embedding-based, Siamese Networks, and attention-based models.

Types of Semantic Similarity:

Knowledge-Based Similarity: Knowledge-based similarity measures how similar two words are based on their meaning, using information from semantic networks. One of the most widely used semantic networks is WordNet, a large database of English words. In WordNet, words are categorized as nouns, verbs, adjectives, or adverbs and grouped into synonym sets, with each group representing a specific concept.

Statistical-Based Similarity: These methods calculate similarity using statistical measures applied to text features or embeddings. Example includes Cosine similarity between vectorized text representations (e.g., TF-IDF, word embeddings).

String-Based Similarity: String-based similarity measures compare the text itself rather than the meaning of the words or sentences. Examples include methods like edit distance and Jaccard similarity.

Language Model-Based Similarity: Language model-based similarity measures use pre-trained language models to understand the meaning of the text. They compare the encoded representations of the text to determine similarity. Examples include models like BERT and Transformers.

To Paraphrase Identification, several tech companies, like Google, Microsoft, etc., released large-scale datasets. Most of the research happened using these datasets. Some of these datasets are Microsoft Research Paraphrase Corpus (MRPC) (Dolan and Brockett, 2005) released by Microsoft, Quora Question Pairs (QQP) (Sharma et al., 2019) released by Quora, Paraphrase Adversaries from Word Scrambling (PAWS) (Zhang et al., 2019a) released by Google. Each of these datasets includes 3 columns: sentence1, sentence2 and label column. The label column indicates whether 2 sentences are paraphrased or not. Based on that, everyone is applying classification matrices to evaluate the result. So precision, recall, F1score and other metrics need to be applied to evaluate the model's performance.

The MRPC (Dolan and Brockett, 2005) comprises 5,801 sentence pairs, with 67% of them being judged semantically equivalent based on human evaluation. Another widely used dataset is the QQP (Quora Question Pairs), which contains questions collected from the Quora platform. Identifying semantically equivalent questions is a crucial task for any question-answering system, including platforms like Quora and Stack Overflow. However, QQP exhibits relatively low lexical overlap between sentence pairs. To address this limitation, (Zhang et al., 2019b) introduced the PAWS dataset, which includes 108,463 sentence pairs with higher lexical overlap. Sentences Involving Compositional Knowledge (SICK) (Marelli et al., 2014) includes a large number of pairs of sentences that have lexical, syntactic and semantic phenomena.

1.2 Related Work

Nicholas Gahman and Vinayak Elangovan have compared 4 types of algorithms: BERT + Cosine Similarity, Multi-Task Deep Neural Network (MT-DNN), XLNet, and Lin Measure + String Similarity, to find out the document databases on multiple datasets MRPC, AFS, SICK-E and SICK-R datasets. They first applied stop word, then word embedding technique (BERT, BoW, Sent2Vec), then, using cosine, Euclidian and Manhattan distances, they found the similarity. They found MT-DNN is the best model for document similarity (Gahman and Elangovan, 2023). Abu Bakar Siddiqur Rahman, along with other researchers, proposed two lightweight models for Paraphrase Identification: linear regression and multilayer perceptron. For mlp they used 3 hidden layers with learning_rate_init=0.01, random_rate=5, activation='relu', and. Also, they used pre-trained models for pair similarity from hugging face but linear regression with noise filtering gives best the result (Bakar et al., 2022).

(Thakur et al., 2021) Nandan thakur, along with other researchers, are working on improving bi-encoders to make them better at scoring pairs of sentences. Bi-encoders encode sentences independently but often require a lot of training data. Augmented SBERT tackles this issue by leveraging a powerful cross-encoder to "soft-label" new sentence pairs. These soft-labeled pairs are then incorporated into the training data for the bi-encoder. This approach significantly improves performance, especially for domain adaptation tasks where labeled data might be scarce (Palivela, 2021). Hemant Palivela used an encoder based fine-tuned Text-To-Text Transfer Transformer (T5) model for paraphrase identification as well as paragraph generation, also used sentence embedding and semantic similarity scores, which helped to achieve paraphrase identification.

(Mohamed and Oussalah, 2020), They propose a hybrid approach for paraphrase identification. They used 2 database wordnets and categorical variation. Firstly, they pre-process the data like text cleaning, tokenizing, part of speech tagging, stop word removal, etc., then apply normalization, which will convert verb, adjective, and adverb categories to nouns. After that, they applied the formula and found the similarity score after combining Wikipedia-based name entity similarity and Catvar-aided wordNet-based similarity.

(Gangadharan Veena et al., 2020), They used different word vectorization techniques for finding paraphrase. Paraphrases also proposed which vectorization method is more efficient. Techniques of word vectorization: Count vectorizer, Term frequency inverse document frequency (TF-IDF), Hashing vectorizer, Word2Vec, Glove, fastText, Embedding from Language Models (Elmo), and BERT. And for document similarity, they used cosine similarity. As per them, fastText is the better method for converting best to a vector and to find its similarity. (Peinelt et al., 2020). They combined topics with BERT for semantic similarity prediction, and it is called tBERT. Latent Dirichlet Allocation (LDA) is used for models. (Ko and Choi, 2020) Proposed paraphrase-BERT. They used MRPC data to fine-tune the pre-trained BERT model, then performed a multitask (MLT) to improve performance. Also, we compared performance between multiple models: the Neural Language Model (NLM), the ARC model, and Bi-CNN-MI.

The existing research primarily focused on either traditional ML models or encoder-based LLMs like Bidirectional Encoder Representations from Transformers (BERT) and T5 models. There hasn't been much study on decoder-based models like Llama and Mistral, especially in terms of using resources efficiently. Also, prompt-based techniques haven't been explored much. This gap shows the need to study both performance and resource optimization of these new models.

1.3 Research Questions

- How well do fine-tuned and prompt-based LLM models perform in identifying paraphrases?
- How do encoder-based models (e.g., ModernBERT), decoder-based models (e.g., Llama3.2-1B, Llama3.2-3B, Mistral-7B, GPT-3.5-turbo, GPT-4o-mini), and encoder-decoder-based models (e.g., T5) compare in terms of paraphrase identification?
- Can resource usage be optimized through fine-tuning without significant performance degradation?
- What fine-tuning strategies can enhance the performance of LLMs on paraphrase identification?
- Which approach, fine-tuned or prompt-based, proves to be more effective for paraphrase identification?

1.4 Aim and Objectives

The primary aim of this research is to conduct a comparative analysis between prompt-based and fine-tuned LLMs for paraphrase identification. Additionally, within fine-tuned models, the study aims to determine whether encoder-based, decoder-based, or encoder-decoder-based architectures provide superior performance in terms of accuracy and computational efficiency. While research has been conducted on paraphrase identification, there is a lack of comprehensive studies comparing prompt-based and fine-tuned approaches across different LLM architectures. Additionally, while model comparisons have been conducted in the past, they primarily used older versions of models that do not reflect the advancements in modern LLMs like Llama-3.3, Mistral-7B, and GPT-4o-mini. This study aims to bridge this gap by evaluating the latest models and identifying the optimal architecture for paraphrase identification. This study utilizes both commercial and open-source LLMs, including prompt-based models (GPT-4o-mini, GPT-3.5-Turbo, Llama-3.2-1B, Llama-3.2-3B, Mistral-7B, T5), fine-tuned decoder-based models (Llama-3.2-1B, Llama-3.2-3B, Mistral-7B), fine-tuned encoder-based models (ModernBERT), and fine-tuned encoder-decoder-based models (T5). This research also investigates the impact of computational resource constraints and optimization techniques such as Low-Rank Adaptation (LoRA) & Quantized Low-Rank Adaptation (QLoRA) to enhance model performance while reducing hardware requirements.

The research objectives are formulated based on the aim of the study, which are as follows:

- To explore and evaluate various paraphrase datasets and select the most suitable one for this study.
- To perform paraphrase identification using both fine-tuned and prompt-based LLM models to assess their effectiveness in real-world applications.
- To conduct a comparative analysis between prompt-based models and fine-tuned models, evaluating their strengths, limitations, and trade-offs for paraphrase identification.
- To optimize resource utilization by applying techniques like LoRA and QLoRA (4-bit quantization) for efficient fine-tuning of large-scale models.
- To analyze the performance of LLM models using key metrics such as accuracy, precision, recall, F1-score, and inference time to provide a holistic evaluation.

- To compare fine-tuned encoder-based models, decoder-based, and encoder-decoder-based models to determine which architecture is more effective for paraphrase identification.

1.5 Scope of the Study

This study focuses on sentence-level paraphrase identification using LLMs. It evaluates both prompt-based and fine-tuned approaches across encoder-based, decoder-based, and encoder-decoder-based architectures. The research specifically utilizes datasets such as MRPC, QQP, and PAWS to ensure diversity in lexical and semantic similarity characteristics. The study also employs quantization techniques 4-bit fine-tuning to achieve optimal model performance with reduced computational costs.

Evaluation metrics such as precision, recall, and F1-score are used to assess model effectiveness, ensuring reliability and generalizability. Additionally, a comparative analysis is conducted to highlight the strengths and limitations of different model architectures in paraphrase identification. While the primary focus is on sentence-level paraphrase detection, paragraph-level and document-level paraphrase detection are beyond the scope of this research. The study primarily uses English-language datasets, but the insights gained may be extended to other languages and domains in future research.

1.5.1 In-Scope

- Analysis of MRPC Dataset for Paraphrase Identification
- **Model Comparison:** Evaluating and comparing prompt-based, fine-tuned decoder-based, fine-tuned encoder-based, and encoder-decoder-based models for paraphrase identification.
- **Models Used:**
    - Prompt-based models: GPT-3.5-turbo, GPT-4o-mini, Llama-3.2-1B, Llama-3.2-3B, Mistral-7B and T5.
    - Fine-tuned decoder-based models: Llama-3.2-1B, Llama-3.2-3B, Mistral-7B.
    - Fine-tuned encoder-based models: ModernBERT.
    - Fine-tuned encoder-decoder-based models: T5
- The research includes both commercial (GPT-3.5-turbo, GPT-4o-mini) and open-source models (Llama-3.2-1B, Llama-3.2-3B, Mistral-7B, ModernBERT, T5) to explore the effectiveness of different architectures.

- **Training Methods:** Utilizing QLoRA and LoRA fine-tuning techniques to enhance model efficiency.

1.5.2 Out of Scope

- Hybrid models that combine multiple architectures (e.g., ensembles of encoder and decoder models) are not considered.
- Any experiments outside the listed models are beyond the scope of this research.
- The study does not focus on multilingual paraphrase identification, as all experiments are conducted in English only.
- Training models from scratch is not performed; only pretrained models with fine-tuning are used.

1.6 Significance of the Study

Paraphrase identification is useful for several natural language processing applications across various domains. Plagiarism detection, question-answering systems, text summarization, and machine translation are the few tasks. Paraphrase identification is used to find out whether two text sequences convey the same meaning or not. In the educational field, to evaluate whether student response aligns semantically with the reference answer. It will improve the functionality of search engines and question-answering systems by better understanding user intent. In the research area, it will help to identify plagiarism in research documents. In summarization, it is used to cut redundant information. Sentences are paraphrased in 2 ways. Sentences are lexical overlap and semantically paraphrased. If sentences lexically overlap without being paraphrased, then most of the models fail to distinguish pairs, like flights from the United States to India and flights from India to the United States. This study emphasizes the critical role of understanding lexical and semantic paraphrase.

A primary contribution of this study is addressing existing research gaps in paraphrase identification. While many studies have focused on encoder-based language models (e.g.: BERT) and limited exploration on decoder-based LLM models (GPT, Llama, Mistral etc.), there is also very limited exploration into resource optimization and lightweight methods. This research will apply quantization techniques on pre-trained language models and try to optimize resource usage. A major advantage of optimizing resource usage makes the findings relevant for deploying NLP models on edge devices and other small devices which have limited computational power. This study not only the understanding of paraphrase identification techniques, but also contributes to other NLP areas like multilingual paraphrase identification, domain-specific applications etc.

1.7 Structure of the Study

Chapter 1: Introduction

This chapter provides an overview of the study, including the background, research question, aim and objectives, and the scope of the research. It also discusses the significance of the study, highlighting its contribution to the field of paraphrase identification using LLM models. Additionally, it defines the in-scope and out-of-scope areas to set clear research boundaries.

Chapter 2: Literature Review

This chapter reviews existing studies and research on paraphrase identification, LLM models, and fine-tuning techniques. It explores the strengths and limitations of encoder-based, decoder-based, and encoder-decoder-based architectures. Furthermore, it examines prior research comparing prompt-based and fine-tuned models, identifying research gaps and justifying the need for this study.

Chapter 3: Research Approach

This chapter outlines the methodology used in the research. It details the dataset selection and description, preprocessing steps, and feature engineering. It also discusses the pre-trained LLM models used and the model training approaches, including prompt-based and fine-tuned training. Additionally, it presents the evaluation metrics for assessing model performance and the computational resources required for experimentation.

Chapter 4: Analysis & Design

This chapter covers exploratory data analysis (EDA), examining the dataset's structure and key patterns. It provides a detailed prompt analysis, discussing different prompt designs and hyperparameter tuning strategies. It also highlights the challenges faced during prompt engineering and fine-tuning, along with an in-depth analysis of fine-tuned models. Various hyperparameter optimization techniques applied to improve model performance are also discussed.

Chapter 5: Results & Discussion

This chapter presents and analyzes the experimental results, comparing the performance of different models using accuracy, precision, recall, and F1-score. It discusses the strengths and weaknesses of prompt-based and fine-tuned models, along with a comparative analysis of encoder-based, decoder-based, and encoder-decoder-based architectures. The findings are interpreted in the context of previous research, emphasizing new insights gained from this study.

Chapter 6: Conclusion & Recommendations

The final chapter summarizes the key findings and contributions of the research. It provides conclusions based on the experimental results and highlights the study's impact on the field of NLP and paraphrase identification. Additionally, it suggests future research directions, including potential improvements in model architectures, training approaches, and resource optimization techniques for real-world applications.

CHAPTER 2

LITERATURE REVIEW

2.1 Introduction

Paraphrase identification is an important task in NLP, with numerous techniques being explored over the years, ranging from traditional machine learning (ML) methods to more advanced deep learning (DL) models. Researchers have used several common approaches to detect paraphrases and check how similar two pieces of text are, utilizing both classical and transformer-based approaches. The methods used in paraphrase identification often include text preprocessing, encoding, embedding techniques, and similarity measures. Below are some of the common techniques employed in this field. References to the studies that used these techniques will be provided later in the document.

2.2 Fundamental Techniques in NLP

2.2.1 Text Preprocessing Techniques

Text preprocessing is a foundation step in NLP tasks. Several researchers (Saeed and Taqa, 2022a) are given the importance of preprocessing for paraphrase identification. Text preprocessing is an important step because raw text data often contains noise, inconsistencies and irrelevant information that can negatively impact performance. And by removing unnecessary characters or text and standardizing text, it will reduce the dimensionality of the data and make it easier for models to process and analyze. To be summarized, text preprocessing is aimed at cleaning and standardizing text data to improve performance, reducing data dimensionality and enhancing model interpretability. Common text preprocessing steps are lowercasing, tokenization, removing punctuation and special characters, removing stop words, stemming and lemmatization, and others.

Using a lowercasing technique, to convert all text to lowercase. This helps standardize the data and reduce the vocabulary size, and words in different cases in sentences are treated as the same. Using tokenization technique, to split text into individual words or sub-words (tokens). Punctuation and special characters often do not carry significant meaning and can be removed to simplify the data. Same as stop words are common words like “the”, “a”, “is” and “to” that do not add much meaning to the text. Removing them can focus the analysis on more informative words. Stemming and lemmatization techniques reduce words to their root and base form (e.g., “running” to “run”). Also, as per requirement, we removed HTML/XML tags.

2.2.2 Text Encoding and Embedding Technique

Researchers (Patil et al., 2023) used various techniques to embed text. There are mostly 2 types: frequency-based and prediction-based embeddings. Frequency-based embeddings, such as TF-IDF, focus on the statistical frequency of words, while prediction-based embeddings, such as Word2Vec and BERT, capture the contextual relationships between words. Once the text is processed, encoding or embedding is an essential step for an NLP task. ML and DL algorithms require numerical input. Text encoding and embeddings convert textual data into a numerical form that these algorithms can process. This includes converting characters into numerical codes, making it easier to handle various languages and special characters. Word embedding captures semantic meanings and relationships between words. For example, in word embeddings, "king" and "queen" might be closer to each other in the vector space than "king" and "apple," reflecting their semantic similarity. Researchers have adopted various encoding and embedding techniques for paraphrase identification, including:

One hot encoding is a technique used to convert text data into a numerical format that can be used by machine learning algorithms. It is simple but suffers from high dimensionality and limited semantic information. A bag-of-words model is a simple document embedding technique based on word frequency. Suppose the whole document as a “bag” of words, rather than a sequence. For example, if we have a vocabulary of 1000 words, then the whole document will be represented by a 1000-dimensional vector, where the vector’s i^{th} entry represents the frequency of the i^{th} vocabulary word in the document. Using this technique, we can embed a whole set of documents and feed them into a variety of different machine learning algorithms. Since this embedding is so simple, it ignores word order and semantic relationships and doesn’t work very well for complex tasks.

Term Frequency-Inverse Document Frequency (TF-IDF) (Qaiser and Ali, 2018) is a numerical statistic that tells us how important a word is in a corpus. It’s a weight that gives us a lot more information than just the number of times a word appears in a document. It combines two metrics: Term Frequency (TF) and Inverse Document Frequency (IDF). Term Frequency measures how frequently a word appears in a document. It is calculated as: TF = (count total number of words in a document) / (check the occurrence of each word in the whole document). Inverse Document Frequency measures how important a word is. It is calculated as: IDF = log (Total number of documents / Number of documents containing the term). The TF-IDF score is obtained by multiplying the TF and IDF values: TF-IDF = TF * IDF. It is easy to calculate, but there are limitations, like it does not capture semantic relationships between words. Performance can be affected by variations in vocabulary and document length.

Word2Vec is a popular word embedding technique developed by a team of researchers at Google led by (Mikolov et al., 2013). It aims to capture the semantic meaning of words by representing them as dense vectors in a high-dimensional space. Word2Vec consists of two main models: Continuous Bag of Words (CBOW) and Skip-Gram. Both models are shallow two-layer neural networks designed to predict words in a given context, but they operate in slightly different ways. The CBOW model predicts the target word from the surrounding context. For example, given the context words "The cat is on the", the model aims to predict the word "mat." Skip-gram predicts the surrounding context words from the target word. For example, given the word "mat," the model aims to predict the context words "The cat is on the." Here's a simplified visual representation of both models:

CBOW: Context Words -> Input Layer -> Hidden Layer -> Output Layer -> Target Word
Skip-Gram: Target Word -> Input Layer -> Hidden Layer -> Output Layer -> Context Words

Doc2Vec, introduced by (Le and Mikolov, 2014) is an extended version of word2vec. While word2vec generates embeddings for individual words, doc2vec creates embeddings for entire documents or paragraphs. This makes it particularly useful for tasks like paraphrase identification, where capturing the contextual meaning of an entire sentence or paragraph is crucial. doc2vec captures both the semantic and syntactic meaning of larger text units, preserving their contextual relationships. It consists of two main models: Distributed Memory (DM) and Distributed Bag of Words (DBOW). The Distributed Memory (DM) model focuses on predicting missing words in a context window while considering a document vector. It helps maintain contextual relationships across the document and captures the flow of meaning.

Distributed Bag of Words (DBOW) model, the document vector is directly predicted from randomly sampled words without considering their context window. DBOW is conceptually similar to the Skip-gram model in word2vec but is applied at the document level. Both models complement each other and are widely used in applications where the semantic understanding of larger text units is essential, such as paraphrase detection, text clustering, and document similarity.

GloVe, introduced by (Pennington et al., 2014), is a method for creating word embeddings that capture both local and global patterns in text. Unlike word2vec, which focuses on nearby words in a context window, glove uses a co-occurrence matrix to measure how often words appear together in a large text corpus. This approach provides a broader understanding of word relationships. Glove minimizes a weighted least squares function to balance the importance of rare and frequent word pairs, resulting in embeddings that capture both meaning and context. These embeddings are widely used in tasks like detecting synonyms, analyzing sentiment, and classifying text because they effectively represent the relationships between words in simple and meaningful ways.

FastText, developed by (Bojanowski et al., 2016) at Facebook, is an extension of Word2Vec that represents words as a combination of character n-grams. Instead of treating each word as a unique entity, FastText breaks words into smaller sub word units, enabling it to capture the meaning of words even when they are rare or misspelled. This approach is especially effective for morphologically rich languages, where words can have many variations. FastText embeddings are powerful for tasks like text classification, machine translation, and semantic similarity, as they can handle out-of-vocabulary words by building representations from their sub word components.

2.2.3 Text Similarity Measure Technique

Cosine similarity is a metric used to measure the similarity between two vectors by calculating the cosine of the angle between them. It is commonly used in text and document analysis to compare word or document embeddings. The cosine similarity value ranges from -1 to 1, where 1 indicates that the vectors are identical in direction, 0 indicates they are orthogonal (unrelated), and -1 indicates they are diametrically opposite. In NLP, cosine similarity is often used to assess the similarity between words, sentences, or documents by comparing their vector representations. This makes it a key tool for paraphrase detection. Most of the researchers used the same.

Jaccard similarity, also known as the Jaccard index, is a measure of similarity between two sets. It is calculated as the size of the intersection divided by the size of the union of the sets. The Jaccard similarity ranges from 0 to 1, where 1 indicates that the sets are identical, and 0 indicates no overlap. In NLP, Jaccard similarity is often used to compare text data by treating each document or sentence as a set of unique words or tokens. This metric is particularly useful for tasks like text clustering, duplicate detection, and paraphrase identification, where the focus is on overlapping content rather than vector-based similarity.

Manhattan distance, also known as L1 distance or taxicab distance, is a distance metric used to calculate the total absolute difference between the coordinates of two points in a grid-like system. It is called Manhattan distance because it resembles the way one would navigate a grid-based city like Manhattan, where movement is restricted to horizontal and vertical paths. For two points, $A(x_1,y_1)$ and $B(x_2,y_2)$ in a 2D plane, the Manhattan distance is calculated as: $|x_2 - x_1|+|y_2 - y_1|$. For points in an n-dimensional space, the formula generalizes to: $\sum_{i=1}^{n}|x_i - y_i|$.

Euclidean distance is the straight-line distance between two points in Euclidean space. It is one of the most commonly used distance metrics, as it measures the shortest path between two points. For two points, $A(x_1,y_1)$ and $B(x_2,y_2)$ in a 2D plane, the Euclidean distance is: $\sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$. For points in an n-dimensional space: $\sqrt{\sum_{i=1}^{n}(x_i - y_i)^2}$.

2.2.4 Transformers Based Embedding Techniques

Several researchers have extensively utilized the BERT & RoBERTa model in their work, leveraging its powerful capabilities for various NLP tasks. Below is the theoretical concept of the BERT & RoBERTa model:

BERT is a Transformer-based architecture developed by Google, designed to excel in natural language understanding tasks. It is built on the encoder part of the Transformer model introduced by(Vaswani et al., 2017). The key innovation of BERT lies in its ability to process text bidirectionally, meaning it can consider the context of a word from both its left and right, enabling a deeper understanding of language compared to traditional unidirectional models. The core of BERT's power comes from its pretraining objectives. The first objective, Masked Language Modeling (MLM), involves randomly masking 15% of tokens in a sentence and training the model to predict these masked words based on the surrounding context. This allows BERT to understand bidirectional dependencies in text. The second objective, Next Sentence Prediction (NSP), trains the model to predict whether a given sentence follows another, helping it captures relationships between sentences. These objectives are key to BERT's effectiveness in tasks involving complex language relationships.

BERT processes input using a combination of embeddings: Token Embeddings, which represent individual words or subwords generated by the WordPiece tokenizer; Segment Embeddings, which distinguish tokens belonging to different sentences in tasks requiring sentence pairs; and Positional Embeddings, which provide positional information about tokens within a sequence. The input format for BERT is structured as [CLS] Sentence A [SEP] Sentence B [SEP], where [CLS] is a special classification token used for tasks like text classification and [SEP] marks the separation between sentences. The architecture of BERT is scalable, with two popular configurations: BERT-Base, consisting of 12 layers, 768 hidden units, and 12 attention heads (110 million parameters), and BERT-Large, with 24 layers, 1024 hidden units, and 16 attention heads (340 million parameters). Each layer incorporates multi-head self-attention mechanisms, allowing the model to focus on different parts of the input simultaneously, and feed-forward neural networks, which process the attended information. Additionally, techniques like layer normalization and residual connections ensure stability and better convergence during training. BERT is pretrained on large corpora such as Wikipedia and BookCorpus, using the Adam optimizer with weight decay for efficient training. The model can then be fine-tuned for specific tasks by adding a task-specific output layer. For example, text classification tasks utilize the output of the [CLS] token, while question-answering tasks leverage token embeddings to predict answer spans. This fine-tuning process requires minimal computational resources compared to pretraining, making BERT highly adaptable to various NLP tasks. Despite its advantages, BERT has limitations. It is resource-intensive, requiring substantial computational power for both pretraining and fine-tuning. The WordPiece tokenization process can sometimes split words unnaturally, affecting interpretability. Additionally, BERT's maximum input length is limited to 512 tokens, which may restrict its application to longer documents. However, its ability to capture bidirectional context, combined with state-of-the-art performance on benchmarks like GLUE and SQuAD, makes BERT a transformative model in NLP.

RoBERTa (Robustly Optimized BERT Pretraining Approach) is an advanced variant of the BERT model introduced by Facebook AI (Liu et al., 2019). It builds upon BERT by refining the pretraining methodology and addressing its limitations, resulting in significant performance improvements across NLP tasks. Unlike BERT, which uses the Next Sentence Prediction (NSP) objective, RoBERTa eliminates NSP entirely, focusing solely on the Masked Language Modeling (MLM) objective. This allows the model to fully leverage bidirectional context without being constrained by NSP's limited contributions to downstream tasks. One of the key improvements in RoBERTa is the use of a much larger pretraining dataset. While BERT was trained on 16GB of text, RoBERTa is trained on over 160GB of diverse text data, including Common Crawl News, OpenWebText, Stories, and BooksCorpus. Additionally, RoBERTa introduces dynamic masking, where the tokens to be masked are chosen randomly at every training epoch, providing more diverse and robust training examples compared to BERT's static masking.

RoBERTa also benefits from increased computational resources. It trains with larger batch sizes, longer sequence lengths, and a higher number of training steps. This extended training enables the model to better capture linguistic patterns and long-range dependencies. Architecturally, RoBERTa retains the same encoder-based Transformer structure as BERT, with configurations such as RoBERTa-Base (12 layers, 768 hidden units) and RoBERTa-Large (24 layers, 1024 hidden units), making it compatible with existing BERT-based applications. The enhancements in RoBERTa make it more robust and effective, achieving superior performance on benchmarks like GLUE, SQuAD, and other natural language understanding tasks. However, these improvements come at the cost of higher computational requirements, as RoBERTa demands significant resources for pretraining. Despite this, its focus on optimizing pretraining and removing redundant objectives has established it as one of the most effective transformer models in NLP.

2.3 Paraphrase Identification using Various NLP Technique

The research on plagiarism detection has seen significant advancements through various approaches and methodologies. The researcher (Saeed and Taqa, 2022a) focuses on developing an effective plagiarism detection system for academic articles. The study utilizes text similarity algorithms and machine learning techniques to detect plagiarized content in documents. The preprocessing steps include tokenization, lowercasing, removal of punctuation and stop words, and lemmatization. The text is encoded using the Term Frequency-Inverse Document Frequency (TF-IDF) method, and the documents are clustered using the K-means algorithm. Cosine similarity is employed to calculate the similarity between the suspicious document and the source corpus. Similarly, the paper text comparison based on semantic similarity (Mhatre et al., 2023) explores the challenge of accurately measuring text similarity, which has applications in plagiarism detection, document clustering, and search engine optimization. The methodology involves preprocessing text data through tokenization, lowercasing, removal of punctuation and stop words, and lemmatization. The text is encoded using vectorization methods such as TF-IDF Vectorizer, Count Vectorizer, Word2Vec, and Sent2Vec. Cosine similarity is used to calculate the similarity between text documents.

The researcher (Al-Jibory and Al-Tamimi, 2021) proposes a hybrid system that incorporates NLP and machine learning (ML) techniques, as well as an external plagiarism detection strategy based on text mining and similarity analysis. The preprocessing steps include normalization, segmentation, stop-word elimination, and lemmatization. The text is analysed using Jaccard and cosine similarity measures to compare the suspect and source documents. In (Kumar and Mehrotra, 2022), the authors evaluate various word embedding techniques and text similarity measures to determine their effectiveness in capturing lexical and semantic similarities between texts. The preprocessing steps include tokenization, lowercasing, removal of punctuation and stop words, and lemmatization. The text is encoded using word embedding techniques such as Word2Vec, FastText, and Doc2Vec. Similarity measures like Jaccard similarity, cosine similarity, Euclidean distance, n-gram similarity, and the longest common substring (LCS) are employed to compare text documents. The researchers (Islam et al., 2023) focus on developing an effective plagiarism detection system for Bengali textual content. The methodology involves preprocessing the text through punctuation removal, stop word elimination, and tokenization. The text is analysed using cosine and Jaccard similarity measures to compare the suspect and source documents. The (Setha and Aliane, 2022) proposes a novel application that utilizes the Doc2Vec model to predict semantic similarity between documents and phrases. The preprocessing steps include tokenization, normalization, and stop-word removal. The text is encoded using the Doc2Vec model to create document and paragraph vectors. Cosine similarity is employed to calculate the similarity between the suspicious document and the source corpus.

Table 2.1: Some NLP based Methods and Results in Paraphrase Detection

Author & Year	Dataset	Models & Method	Performance Measure
(Al-Jibory and Al-Tamimi, 2021)	PAN-PC-11	Text Preprocessing with average of Jaccard and Cosine Similarity	Accuracy: 0.96, Recall: 0.86, Precision: 0.86, Plag-Det score: 0.86
(Setha and Aliane, 2022)	PAN-PC-2009 and Ara-Plag corpora	Doc2vec-Arabic	Precision: 0.8, Recall: 0.9, F1-score: 0.85
		Doc2vec-English	Precision: 0.9, Recall: 0.99, F1-score: 0.94
(Saeed and Taqa, 2022b)	PAN-PC-11	Siamese LSTM + word2vec	Precision: 0.924, Recall: 0.917, F1-score: 0.816
	Webis-CPC-11	Siamese LSTM + word2vec	Precision: 0.902, Recall: 0.896, F1-score: 0.793

2.4 Paraphrase Identification using Machine Learning Technique

The paper titled "Machine Learning Models for Paraphrase Identification and its Applications to Plagiarism Detection" (Hunt et al., 2019) focuses on the challenging task of paraphrase identification in NLP. The authors explore various machine learning models, including Logistic Regression, Support Vector Machines (SVM), and different architectures of Neural Networks, with a particular emphasis on Recurrent Neural Networks (RNNs) and Long Short-Term Memory (LSTM) networks. The methodology involves preprocessing text data through tokenization, normalization, and stop-word removal, followed by encoding using One-Hot Encoding and Word2Vec embeddings. The models are trained and evaluated on the Quora Duplicate Question dataset, consisting of over 400,000 question pairs. Key findings indicate that the LSTM model outperforms other models, achieving the highest accuracy in paraphrase identification. The study concludes that paraphrase identification can significantly enhance plagiarism detection systems by identifying semantically equivalent content, even when the syntax differs.

The researchers (Vasuteja et al., 2024) addresses the significant issue of plagiarism in academic and professional settings, emphasizing the need for effective detection methods to maintain originality and integrity. The study explores various machine learning models, including SVM, Naive Bayes, Random Forests, CNNs, and RNNs, with a focus on their accuracy and performance in detecting plagiarized content. The methodology involves preprocessing text data through tokenization, lowercasing, removal of punctuation and stop words, and stemming. The text is then analysed using similarity measures such as cosine similarity, Jaccard similarity, Levenshtein distance, and edit distance. The proposed system evaluates paraphrasing and rewriting techniques in addition to literal copying, providing a comprehensive approach to plagiarism detection. The study concludes that the proposed system can significantly enhance plagiarism detection by providing detailed reports highlighting plagiarized portions and their original sources. The researchers (Chavan Hiten et al., 2021) study explores machine learning models, including the TF-IDF Vectorizer and cosine similarity measures, to detect plagiarized content. The methodology involves preprocessing text data through tokenization, lowercasing, removal of punctuation and stop words, and stemming. The text is analyzed using cosine similarity to calculate the similarity between the suspicious document and the source corpus. The proposed system evaluates paraphrasing and rewriting techniques in addition to literal copying, providing a comprehensive approach to plagiarism detection. Key findings indicate that the system achieves high accuracy in identifying plagiarized content. This research is relevant to my work as it offers a robust approach to plagiarism detection, utilizing advanced NLP and machine learning techniques.

2.5 Paraphrase Identification using Deep Learning Models

This task (Kubal and Nimkar, 2018) is essential for applications such as information extraction, question answering, and plagiarism detection. The study proposes a hybrid deep learning architecture that combines Long Short-Term Memory (LSTM) networks and Convolutional Neural Networks (CNN) to capture both sequential dependencies and feature extraction capabilities. The methodology involves preprocessing text data through tokenization, normalization, stop-word removal, and lemmatization. The text is then encoded using the context2vec model, which generates 300-dimensional word vectors. The hybrid model uses LSTM to handle variable-length sentences and CNN to process fixed-size similarity matrices derived from cosine similarity measures. The proposed model was evaluated on the MRPC and the Quora Question Pairs dataset, demonstrating superior performance with an accuracy of 79.88% when using context2vec embeddings. Key findings indicate that the hybrid model effectively captures semantic relationships between sentences, outperforming traditional methods and other deep learning models. The study (Aziz et al., 2019) aims to develop an effective plagiarism detection system by leveraging deep learning techniques, specifically Recurrent Neural Networks (RNN) and Long Short-Term Memory (LSTM) networks. The methodology involves preprocessing the text through tokenization, normalization, stop-word removal, and lemmatization. The text is then encoded using the Word2Vec model to create word vectors, which represent the semantic and syntactic similarity of words. Sentences are represented as vectors by aggregating the word vectors of each word in the sentence. The Siamese network architecture, particularly the Manhattan LSTM, is used to measure the similarity between sentence pairs. The proposed model was tested on a dataset of 800,000 sentence pairs and demonstrated an accuracy of 99% on training data and 82.4% on new data. Key findings indicate that the amount of training data significantly impacts the model's performance, with larger datasets yielding higher accuracy.

The researchers (Aziz et al., 2024) aim to develop an effective plagiarism detection system by leveraging deep learning techniques, specifically Recurrent Neural Networks (RNN) and Long Short-Term Memory (LSTM) networks. The methodology involves preprocessing the text through tokenization, normalization, stop-word removal, and lemmatization. The text is then encoded using the Word2Vec model to create word vectors, which represent the semantic and syntactic similarity of words. Sentences are represented as vectors by aggregating the word vectors of each word in the sentence. The Siamese network architecture, particularly the Manhattan LSTM, is used to measure the similarity between sentence pairs. The proposed model was tested on a dataset of 800,000 sentence pairs and demonstrated an accuracy of 99% on training data and 82.4% on new data. Key findings indicate that the amount of training data significantly impacts the model's performance, with larger datasets yielding higher accuracy. This research is relevant to my work as it provides a comprehensive approach to plagiarism detection, utilizing advanced NLP and deep learning techniques. The researchers (Saeed and Taqa, 2022b) propose a deep learning framework for detecting text plagiarism using a network of Siamese Long Short-Term Memory (LSTM) and word embeddings. The methodology involves preprocessing text data through tokenization, lowercasing, removal of punctuation and stop words, and lemmatization. The text is then encoded using Word2Vec and GloVe models to create word embeddings. The Siamese LSTM network is employed to measure the similarity between text pairs, utilizing a hybrid of Manhattan distance and cosine similarity measures. The proposed model was evaluated on the PAN-PC-11 and Webis-CPC-11 datasets, demonstrating high accuracy in detecting plagiarized content. Key findings indicate that the Word2Vec embeddings produced the best detection scores, with an F1-measure of 0.816, recall of 0.91, and precision of 0.924 for the PAN-PC-11 corpus, and an F1-measure of 0.793, recall of 0.852, and precision of 0.902 for the Webis-CPC-11 corpus.

The researchers (Saqaabi et al., 2022) propose a novel approach that leverages machine learning (ML) and deep learning (DL) techniques to measure the semantic similarity of natural language in longer texts. The methodology involves preprocessing text data through tokenization, normalization, stop-word removal, and lemmatization. The text is then encoded using TF-IDF, Bleu metric, N-gram overlap, and Word2Vec as features. The Support Vector Machine (SVM) classifier is employed to detect paraphrases. The proposed model was evaluated on two benchmark datasets, Webis-CPC-11 and MSRP, demonstrating superior performance in detecting paraphrases at the paragraph level compared to sentence-level approaches. Key findings indicate that considering text at paragraph length is more appropriate for ML and DL approaches, with the proposed method outperforming existing work on the Webis-CPC-11 dataset. The study concludes that the length of the text significantly affects the accuracy of paraphrase identification models, with mid-length texts (paragraphs) providing the best results. (Zhang et al., 2017).The study proposes a novel approach that integrates syntactic and semantic information using a Convolutional Neural Network (CNN) model to identify paraphrases. The methodology involves several steps: first, words are mapped to a low-dimensional space using word embeddings to obtain their semantic representation. Next, the vectors representing phrases and sentences are obtained using a parse tree combined with Recursive Autoencoder (RAE) to capture syntactic and semantic features. Three types of similarity matrices are created using these vector representations to capture the complex interrelationships between two sentences.

These similarity matrices are then used as input to the CNN to predict the probability of two sentences being paraphrases. The CNN architecture includes multiple convolutional layers, dynamic pooling, and a final prediction layer. The proposed model was evaluated on the MRPC, which consists of 5801 sentence pairs. The results demonstrate that the model achieves superior performance compared to traditional methods, with an accuracy of 77.51% and an F1 score of 84.38%. Key findings indicate that the integration of syntactic and semantic information, along with the use of CNN, significantly enhances the model's ability to identify paraphrases. This research is relevant to my work as it provides a comprehensive approach to paraphrase identification, utilizing advanced NLP and deep learning techniques. The insights gained from this study can be applied to enhance the accuracy and efficiency of plagiarism detection systems, supporting the integrity and originality of scholarly work. The study's focus on combining different similarity measures and preprocessing steps aligns with my research objectives, making it a valuable reference for my thesis.

The study proposes by (Kuksa and Polyakov, 2024) a neural network-based system for detecting plagiarism in academic and research works. The methodology involves several key stages: dataset creation, model development, and training. The dataset is generated by parsing open resources and aggregating scientific publications, and the text is processed in groups to form a balanced dataset. The neural network architecture is a fully-connected model with two topologies: one with a single hidden layer and another with two hidden layers. The text is vectorized using pre-trained models from the spaCy library, which provides fixed-length vectors for text representation. The neural network is trained on a dataset of 45000 elements, with 20% reserved for testing. The model's performance is evaluated using metrics such as Mean Absolute Error (MAE), Mean Square Error (MSE), and the coefficient of determination (R2). The results indicate that the model with two hidden layers outperforms the one with a single hidden layer, achieving an MAE of 9.38, an MSE of 164.22, and an R2 of 0.71. These findings demonstrate the effectiveness of the proposed system in accurately detecting plagiarism, making it a valuable tool for promoting academic integrity and ensuring educational quality. The study concludes that the developed system can significantly reduce the time spent by instructors on checking students' work, thereby freeing up resources for better instruction and student support. The paper titled "Legal Document Similarity Matching Based on Ensemble Learning" by (Fan et al., 2024) addresses the challenge of matching similar legal cases to improve the accuracy and consistency of legal decision-making. The study proposes an innovative approach that employs ensemble learning with multiple models to enhance the prediction of legal case similarity. The methodology involves two sub-networks: a similarity representation subnetwork and a binary classification judgment subnetwork. The similarity representation subnetwork is trained using contrastive learning, focusing on semantic the similarity between sample features to distinguish between dissimilar samples and reduce the distance between similar ones. The binary classification judgment subnetwork integrates sample pairs to facilitate feature interaction between text pairs during extraction. The training of these two subnetworks employs different information processing and optimization loss, allowing ensemble learning to capitalize on the strengths of both models. Data augmentation techniques, such as commutativity, anti-symmetry, reflexivity, and their combination, are used to increase the diversity and size of samples, improving the model's performance during training. The results indicate that the proposed method achieves an accuracy of 74.53% on the test set, outperforming other existing methods on the public dataset CAIL2019-SCM. The study concludes that the developed system can significantly enhance the accuracy and efficiency of legal case similarity matching, making it a valuable tool for promoting fairness, consistency, and accuracy in legal decision-making.

The researchers (Omi et al., 2021) address the challenge of identifying multiple authors of source code, which is crucial for tasks such as software plagiarism detection, authorship disputes, and malicious code detection. The study proposes a novel approach that leverages machine learning (ML) and deep learning (DL) techniques to identify multiple authors by analyzing their coding styles. The methodology involves several key stages: dataset collection, feature extraction, and classification. The dataset, collected from GitHub, includes 3021 Java source code files from 40 authors. The feature extraction process uses the Abstract Syntax Tree (AST) to generate path-based structural representations of the source code, which are then converted into numerical vectors using the code2seq model. This model provides a syntactic representation of the source code by encoding each AST path with a bi-directional Long Short-Term Memory (LSTM) network. The study employs three classifiers: Deep Neural Network (DNN), Support Vector Machine (SVM), and LSTM. The DNN consists of three fully-connected layers with ReLU activation functions, two dropout layers, and an output layer with a softmax activation function. The SVM classifier uses a penalty parameter (C) to handle the error factor, while the LSTM classifier includes 100 memory units. The training process involves splitting the dataset into training, development, and test sets, with 80% of the data used for training, 10% for development, and 10% for testing. The classifiers are trained to analyze the coding patterns of specific class labels, representing the authors. The results indicate that the DNN classifier outperforms the other models, achieving a classification accuracy of 96.7%, while the SVM and LSTM classifiers achieve accuracies of 87.3% and 94%, respectively. The study concludes that the proposed method effectively identifies multiple authors of source code, making it a valuable tool for promoting academic integrity and ensuring educational quality. This research is relevant to my work as it provides a comprehensive approach to plagiarism detection, utilizing advanced NLP and machine learning techniques. The insights gained from this study can be applied to enhance the accuracy and efficiency of plagiarism detection systems in academic settings, supporting the integrity and originality of scholarly work. The study's focus on combining different similarity measures and preprocessing steps aligns with my research objectives, making it a valuable reference for my thesis.

The researcher (Li et al., 2020b) proposes a novel approach that integrates deep learning techniques, specifically leveraging the ResNeXt architecture, to improve the performance of paraphrase identification models. The methodology involves several key stages: data preprocessing, model development, and training. The data preprocessing includes data cleaning, word segmentation, and word vectorization. The neural network architecture is a fully-connected model with multiple convolution kernels of different sizes to extract features from word sequences of varying lengths. The model consists of convolutional layers, residual units, pooling layers, and fully connected layers. The input text is transformed into a k-dimensional word vector, and the convolutional layers extract ternary features of the text, capturing key issues such as improper word collocation and reversed word order. The residual units, inspired by Inception Net, use group convolutions to extract different features from different subspaces in feature maps, and concatenate all group outputs into a larger space. The pooling layer uses overlapping pooling with a step size of 2 and a window size of 3x1, and the K-max algorithm is employed to select the largest k features for final classification. The fully connected layers connect the feature matrix to the classification output. The model was evaluated on the MRPC and a Chinese dataset provided by 360 Search. The results indicate that the proposed model achieves an accuracy of 85.4% on the MSRPC dataset, outperforming other state-of-the-art models such as Text-CNN, ResNet-34, and GRU. The study concludes that the developed system can significantly enhance the accuracy and efficiency of paraphrase identification, making it a valuable tool for promoting academic integrity and ensuring educational quality.

The researcher (Wang et al., 2018) proposes a novel approach that combines neural networks and keyword-based methods to measure the similarity between sentences at both lexical and semantic levels. The methodology involves two main modules: a neural network framework and a keyword-based framework. The neural network framework employs a Siamese Bidirectional Long Short-Term Memory (Bi-LSTM) model to capture long-term dependencies and semantic similarities between sentences. The Bi-LSTM model consists of two weight-sharing networks that process sentence pairs and generate vector offsets to represent the relationship between sentences in vector space. The keyword-based framework extracts keywords from sentences using an Automatic Keyword Extraction module and measures sentence similarity based on the co-occurrence of keywords. The final similarity score is obtained by merging the outputs of both modules. The model was evaluated on the MRPC and the SICK dataset, demonstrating state-of-the-art performance with an accuracy of 80.2% and an F1 score of 86.4% on MSRP, and an accuracy of 87.1% on the SICK dataset. Key findings indicate that the integration of lexical and semantic information significantly enhances the model's ability to recognize paraphrases, outperforming existing methods. The study concludes that the proposed approach provides a robust and efficient solution for paraphrase recognition, making it a valuable tool for various NLP tasks. (Gontumukkala et al., 2022). The study employs various NLP and Deep Learning (DL) techniques to tackle these issues. The methodology involves preprocessing the text data through stop word removal, punctuation removal, misspelled word corrections, and lemmatization. Five different word embeddings—Word2Vec, GloVe, FastText, Doc2Vec, and Paraphrase-MiniLM-L6-v2—are used to convert sentences into vectors. For question pairs identification, the study utilizes a Siamese Manhattan Long Short-Term Memory (MaLSTM) model, which calculates the Manhattan distance between sentence pairs to determine their similarity. For insincere question classification, a BiLSTM (Bidirectional Long Short-Term Memory) combined with a BiGRU (Bidirectional Gated Recurrent Unit) and an attention mechanism is employed. The models are trained and evaluated using datasets from Kaggle, with hyperparameter tuning performed to optimize performance. The results indicate that the Paraphrase-MiniLM-L6-v2 embedding combined with the Siamese MaLSTM model achieves the highest accuracy of 90% and an F1 score of 0.89 for question pairs identification. For insincere question classification, the FastText embedding combined with the BiLSTM and BiGRU models achieves the highest accuracy of 96% and an F1 score of 0.82. The study concludes that the proposed models significantly improve the accuracy and efficiency of detecting duplicate and insincere questions on Quora, making them valuable tools for enhancing user experience and maintaining the quality of the platform.

Table 2.2: Some NLP+DL based Methods and Results in Paraphrase Detection

Author & Year	Dataset	Models & Method	Performance Measure
(Kubal and Nimkar, 2018)	MRPC + Quora Question Pairs	Word2vector + Hybrid Deep Learning (LSTM+CNN)	Accuracy: 0.7766, Precision: 0.83271, Recall: 0.8138, F1-score: 0.8231
		GloVe + Hybrid Deep Learning (LSTM+CNN)	Accuracy: 0.7849, Precision: 0.8393, Recall: 0.8203, F1-score: 0.8297
		Context2vec + Hybrid Deep Learning (LSTM+CNN)	Accuracy: 0.7988, Precision: 0.8503, Recall: 0.8310, F1-score: 0.8406
(Hunt et al., 2019)	Quora Question Pairs	LSTM	Accuracy: 0.8
(Li et al., 2020a)	MRPC	Bi-LSTM	Accuracy: 86.2, F1-score: 90.1
(Wang et al., 2018)	MRPC	Bi-LSTM+Similarity Distance	Accuracy: 0.80, F1-score: 0.86
	SICK		Accuracy: 0.87
(Li et al., 2020a)	MRPC	Bi-LSTM	Accuracy: 86.2, F1-score: 90.1
(Zhang et al., 2017)	MRPC	CNN	Accuracy: 0.77, F1-score: 0.84
(Li et al., 2020b)	MRPC	ResNeXt	Accuracy: 0.85

2.6 Paraphrase Identification using Transformer based Encoder Model

(Zhang et al., 2022) employs a deep Siamese network architecture, leveraging a large-scale pre-trained BERT model to represent text as word vectors. The methodology involves preprocessing the text through tokenization, normalization, stop-word removal, and lemmatization. The text is then encoded using BERT to obtain contextual semantic features. The Siamese network is designed to map matched text pairs into the same parameter matrix space, ensuring uniform representation. Additionally, the model incorporates a multi-head self-attention mechanism to fuse text pair vectors for accurate semantic alignment and similarity measures. The experimental results demonstrate the effectiveness of the proposed model in identifying and detecting plagiarized text. The model was evaluated on three datasets: MSRP, SNLI, and SemEval, showing superior performance compared to other methods. Key findings indicate that the proposed model achieves high accuracy, recall, precision, F1-measure, and G-means, making it a robust solution for plagiarism detection. The study concludes that the integration of BERT and Bi-LSTM networks, along with the text semantic interaction mechanism, significantly enhances the model's ability to detect plagiarism.

The researchers (Latina et al., 2024) addresses the challenge of detecting paraphrased and translated texts, which traditional plagiarism detection tools struggle with. The study proposes a system that integrates NLP techniques, specifically Word2Vec and BERT, to detect plagiarism in paraphrased texts. The methodology involves preprocessing the text through tokenization, normalization, stop-word removal, and lemmatization. The text is then encoded using Word2Vec and BERT models to create word embeddings that capture semantic and syntactic similarities. The system employs cosine similarity and the Longest Common Subsequence (LCS) techniques to measure the similarity between the suspect and source documents. The hybrid model achieved 93% accuracy in detecting paraphrased plagiarism. The system was evaluated using the ISO 25010 software quality model, which showed excellent results in Functional Suitability, Performance Efficiency, Usability, and Reliability, with mean scores of 4.29, 4.29, 4.59, and 4.34, respectively. Key findings indicate that the integration of Word2Vec and BERT models significantly enhances the system's ability to detect paraphrased plagiarism, providing a robust and efficient solution (Ko and Choi, 2020). This task is essential for applications such as information retrieval, text mining, plagiarism detection, query ranking, query response, and document summarization. The study proposes a Paraphrase-BERT model that leverages the BERT architecture, enhanced with Whole Word Masking (WWM) and Multi-Task Learning (MTL) techniques. The methodology involves fine-tuning the pre-trained BERT model with the MRPC data and incorporating WWM to improve the model's performance. Additionally, the model is trained on a more challenging Question Answering (QA) task using the SQuAD dataset before fine-tuning with MRPC data to enhance the paraphrase identification task. The results demonstrate that the Paraphrase-BERT model significantly outperforms previous deep neural network models, achieving an 11.11% absolute accuracy improvement and a 7.88% absolute F1 score improvement over existing models. The study concludes that the combination of WWM and MTL techniques with BERT leads to superior performance in paraphrase identification tasks.

The researchers (Rosu et al., 2020) propose a novel approach that leverages state-of-the-art deep learning techniques, specifically focusing on the use of BERT and RoBERTa models. The methodology involves preprocessing text data through tokenization, normalization, stop-word removal, and lemmatization. The text is then encoded using pre-trained models from BERT and RoBERTa, which are fine-tuned using the STS benchmark dataset to obtain relevant results for plagiarism detection. The system employs cosine similarity, Manhattan distance, Euclidean distance, and dot product similarity measures to evaluate the similarity between the suspect and source documents. The experimental results demonstrate that the BERT model outperforms GloVe and RoBERTa in terms of accuracy and relevance, achieving superior performance in detecting plagiarized content. Key findings indicate that the BERT model provides a better ranking of documents based on their similarity to the query document, with a significant improvement in context understanding and document ranking. The paper (Li et al., 2020a) proposes a semi-supervised deep learning framework that utilizes a neural encoder with a word-by-word attention mechanism to reason equivalence or contradiction over pairs of words, phrases, and sentences. The methodology involves a two-stage training procedure: first, a language modeling objective is used to learn initial parameters on an unlabeled corpus of over one million sentence pairs, followed by supervised training to adapt these parameters to a specific classification task with labeled data. The neural encoder consists of three layers: look-up, hidden, and attention layers, with Bi-LSTMs (Bidirectional Long Short-Term Memory) used for conditional encoding of sentences. The attention mechanism learns semantic relevance scores for words in the second sentence based on the first sentence's context. The proposed model was evaluated on the MRPC and the Sentences Involving Compositional Knowledge (SICK) datasets. Key findings indicate that the model achieves absolute improvements in accuracy of 7.6% and an F1 score of 5.4% on MRPC, and Pearson's r of 4.5% and Spearman's ρ of 5.1% on SICK, compared to previous neural network models.

The researchers (Eppa and Murali, 2021) address the challenge of detecting plagiarism in programming assignments, particularly focusing on multisource plagiarism where code is copied from multiple sources. The study proposes a novel approach that leverages machine learning and deep learning techniques to detect plagiarism in C programming assignments. The methodology involves several key stages: splitting programs into functions, feature extraction, vector generation, clustering, and similarity measurement. The programs are first split into their corresponding functions using the pycparser library, which parses C code into an Abstract Syntax Tree (AST). Features are then extracted from each function using a transformer network called CodeBERTa, which is a pre-trained model fine-tuned on the CodeSearchNet dataset. The vectors representing the functions are generated using the transformer network and stored in a nested dictionary for quick access. The DBSCAN clustering algorithm is used to find groups of similar submissions based on the extracted features, using point density in the vector space to cluster submissions. The final similarity score is obtained by mapping the various functions and considering the similarity scores of each function along with their lengths. The results indicate that the proposed approach is effective in detecting multisource plagiarism, outperforming existing tools like MOSS and JPlag, which struggle with code obfuscation and one-to-one comparisons. The study concludes that the developed system can significantly enhance the accuracy and efficiency of plagiarism detection in programming assignments, making it a valuable tool for promoting academic integrity.

Table 2.3: Some Transformer based Methods and Results in Paraphrase Detection

Author & Year	Dataset	Models & Method	Performance Measure
(Ko and Choi, 2020)	MRPC	BERT-large + WWM + Multi model	Accuracy: 89.21, F1-score: 92.28
(Saqaabi et al., 2022)	MRPC	SBERT	Accuracy: 72, F1-score:82
	Webis-CPC-11		Accuracy: 64, F1-score:63

(Zhang et al., 2022)	MRPC	BERT+Bi-LSTM	Accuracy: 0.919, Precision: 0.938, Recall: 0.956, F1-score: 0.943
	SNLI		Accuracy: 0.912, Precision: 0.865, Recall: 0.901, F1-score: 0.891
	SemEval2014		Accuracy: 0.855, Precision: 0.977, Recall: 0.878, F1-score: 0.919
(Fan et al., 2024)	CAIL2019-SCM	Multi BERT Network	Accuracy: 0.74
(Gontumukkala et al., 2022)	Quora Question Pairs	ParaphraseMiniLM-L6-v2 + Siamese MaLSTM	Accuracy: 0.90, Precision: 0.85, Recall: 0.94, F1-score: 0.89

2.7 Key Datasets Used for Paraphrase Identification Studies

2.7.1 PAN-PC-11

The PAN-PC-11 dataset is part of the PAN (Plagiarism Analysis, Authorship Identification, and Near-Duplicate Detection) competition held in 2011. It was designed for evaluating plagiarism detection systems and contains artificially created and real plagiarized text fragments. The dataset includes suspicious and source documents in multiple languages, annotated with the types and locations of plagiarized passages. This makes it useful for studying text alignment and similarity detection algorithms.

2.7.2 PAN-PC-2009

Similar to PAN-PC-11, the PAN-PC-2009 dataset was introduced during the 2009 PAN competition. It primarily focuses on plagiarism detection and includes both simulated and real cases of plagiarism. The dataset includes external plagiarism, where plagiarized sections reference external documents, and intrinsic plagiarism, which is detectable through stylistic inconsistencies within a single document. Researchers use this dataset to test and benchmark plagiarism detection tools.

2.7.3 Ara-Plag Corpora

The Ara-Plag corpora is a dataset specifically designed for plagiarism detection in the Arabic language. It addresses the unique linguistic challenges of Arabic, such as rich morphology and free word order. The dataset contains Arabic documents with varying levels of plagiarism, including direct copy, paraphrasing, and idea plagiarism. It is a valuable resource for developing and evaluating plagiarism detection systems for Arabic text.

2.7.4 Quora Question Pairs (QQP)

The Quora Question Pairs dataset contains over 400,000 pairs of questions, annotated with labels indicating whether the questions are semantically equivalent. It is widely used for paraphrase detection, as it focuses on identifying pairs of questions with the same intent but different wording. The dataset is popular for training and evaluating models in tasks like semantic similarity and duplicate detection.

2.7.5 Microsoft Research Paraphrase Corpus (MRPC)

The MRPC dataset consists of 5,801 sentence pairs collected from online news sources, with annotations specifying whether the pairs are semantically equivalent. It is one of the standard datasets for evaluating paraphrase detection models. The dataset emphasizes real-world variability in phrasing, making it an excellent benchmark for natural language understanding tasks.

2.7.6 Webis-CPC-11

The Webis-CPC-11 (Webis Crowdsourcing Paraphrase Corpus) dataset is a collection of paraphrased text generated through crowdsourcing. It contains sentence pairs labeled as paraphrases or non-paraphrases, focusing on linguistic diversity and varying similarity levels. The dataset is often used to evaluate models for paraphrase detection and text similarity tasks.

2.7.7 SNLI (Stanford Natural Language Inference)

The SNLI dataset contains 570,000 sentence pairs, with each pair annotated for entailment, contradiction, or neutral relationships. It is widely used for natural language inference tasks, which involve determining the logical relationship between two sentences. While it is not specifically a plagiarism detection dataset, its emphasis on semantic relationships makes it useful for paraphrase detection and related tasks.

2.7.8 SemEval 2014

SemEval (Semantic Evaluation) is an annual series of NLP challenges. The 2014 edition introduced several tasks, including one on semantic textual similarity (STS). The STS task involved datasets with sentence pairs labeled with similarity scores. These datasets are used to evaluate the performance of models in capturing semantic similarity, which is relevant to plagiarism and paraphrase detection.

2.7.9 CAIL201-SCM

CAIL201-SCM (Chinese AI and Law) is a legal-domain dataset focused on semantic matching of case elements. It includes sentence pairs with labels indicating whether the sentences are semantically similar. The dataset is widely used for paraphrase detection and textual entailment in legal texts. It addresses domain-specific challenges such as formal language and precise semantics.

2.7.10 SICK (Sentences Involving Compositional Knowledge)

The SICK dataset contains 10,000 English sentence pairs labeled for relatedness (semantic similarity) and entailment (logical relationship). It focuses on capturing the compositional semantics of sentences and is often used for tasks such as paraphrase detection, semantic similarity, and natural language inference. The dataset emphasizes linguistic variability and complex semantic relationships.

2.8 Summary

The study by Al-Jibory and Al-Tamimi (2021) focuses on plagiarism detection in scientific papers using a hybrid system. Their approach combines text preprocessing with Jaccard and Cosine Similarity to achieve high accuracy (0.96) and a plagiarism detection score of 0.86 in the PAN-PC-11 dataset. Similarly, Setha and Aliane (2022) utilize the Doc2vec model for plagiarism detection on both the PAN-PC-2009 and Ara-Plag corpora. Their method achieves an F1 score of 0.85 for Arabic text and 0.94 for English text, showcasing the model's adaptability to different languages. Hunt et al. (2019) applied LSTM-based models on the Quora Question Pairs dataset for paraphrase identification, achieving an accuracy of 0.8. Meanwhile, Kubal and Nimkar (2018) propose a hybrid deep learning architecture combining LSTM and CNN with different word embeddings, including Word2Vec, GloVe, and Context2vec. Using the MRPC and Quora datasets, their best performance was observed with Context2vec, achieving an accuracy of 0.7988 and an F1 score of 0.8406. Saeed and Taqa (2022b) implement a Siamese LSTM with Word2Vec embeddings for textual plagiarism detection, achieving high precision (0.924) and recall (0.917) on the PAN-PC-11 dataset and comparable results on the Webis-CPC-11 dataset.

In Ko and Choi's (2020) work, a BERT-based model with Whole Word Masking (WWM) and multi-task learning achieves impressive results on the MRPC dataset, with an accuracy of 89.21

and an F1 score of 92.28. Similarly, Saqaabi et al. (2022) utilize SBERT for paraphrase identification on MRPC, achieving an F1 score of 82. In the Webis-CPC-11 dataset, their method performs slightly less effectively, with an F1 score of 63. Li et al. (2020a) introduce a semi-supervised Bi-LSTM model for paraphrase identification, achieving a high accuracy of 86.2 and an F1 score of 90.1 on the MRPC dataset. Zhang et al. (2022) develop a deep Siamese network combining BERT and Bi-LSTM, achieving excellent results across multiple datasets, including MRPC (F1: 0.943), SNLI (F1: 0.891), and SemEval2014 (F1: 0.919). Zhang et al. (2017) explored CNN-based paraphrase identification, achieving an accuracy of 0.77 and an F1 score of 0.84 on MRPC. The work by Fan et al. (2024) focuses on legal document similarity using a multi-BERT network, attaining an accuracy of 0.74 on the CAIL2019-SCM dataset. Li et al. (2020b) utilize the ResNeXt architecture for paraphrase identification on the MRPC dataset, achieving an accuracy of 0.85. Wang et al. (2018) propose a combined Bi-LSTM and keyword-based framework for paraphrase recognition, achieving an accuracy of 0.80 and an F1 score of 0.86 on MRPC, and a higher accuracy of 0.87 on the SICK dataset. Finally, Gontumukkala et al. (2022) address Quora question pair identification using the Paraphrase-MiniLM-L6-v2 embedding with Siamese MaLSTM, achieving an accuracy of 0.90 and an F1 score of 0.89, demonstrating the model's efficacy in identifying duplicate and insincere questions.

CHAPTER 3

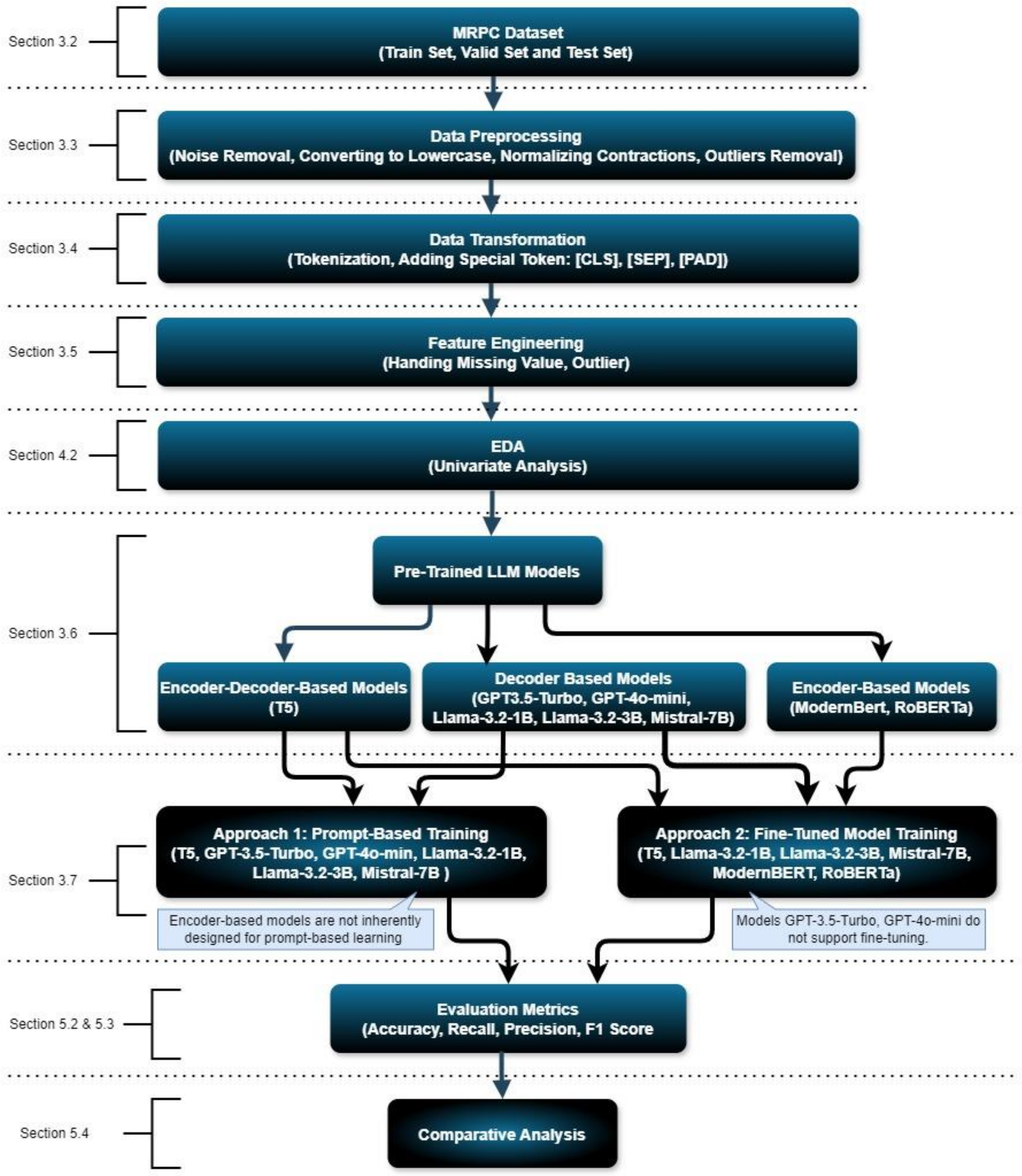
Research Approach

3.1 Introduction

The chapter is structured to provide a comprehensive understanding of the data, models, and techniques used in the study. The research methodology is divided into several key components, including data description and preprocessing, model architectures, fine-tuning techniques, prompt engineering, and evaluation metrics. Each component is designed to ensure a systematic and well-informed approach to achieving the research objectives.

The chapter begins with a detailed discussion of the MRPC, including its structure, preprocessing requirements, and transformation techniques. The dataset consists of 5801 sentence pairs, with 67% paraphrase pairs and 33% non-paraphrase pairs. Preprocessing steps, such as text normalization, tokenization, and noise removal, are applied to ensure the dataset is clean and consistent. The chapter then explores the general transformer architecture and specific architectures like Llama and Mistral, which are used as the foundation for both fine-tuned and prompt-based approaches.

The chapter also delves into fine-tuning techniques, including LoRA and QLoRA, which are used to adapt pre-trained models to the paraphrase identification task. Prompt engineering and in-context learning are discussed as part of the prompt-based approach, highlighting the challenges and strategies for designing effective prompts. The proposed methodology combines model-based fine-tuning and prompt-based learning, with a focus on optimizing performance and resource usage. Finally, the chapter concludes with a discussion of evaluation metrics, such as accuracy, precision, recall, and F1 score, which are used to assess the performance of the models. This comprehensive methodology ensures a robust and well-informed approach to paraphrase identification, providing a solid foundation for the experimental setup and evaluation.



Section 3.2
MRPC Dataset
(Train Set, Valid Set and Test Set)
Section 3.3
Data Preprocessing
(Noise Removal, Converting to Lowercase, Normalizing Contractions, Outliers Removal)
Section 3.4
Data Transformation
(Tokenization, Adding Special Token: [CLS], [SEP], [PAD])
Section 3.5
Feature Engineering
(Handing Missing Value, Outlier)
Section 4.2
EDA
(Univariate Analysis)
Section 3.6
Pre-Trained LLM Models
Encoder-Decoder-Based Models (T5)
Decoder Based Models (GPT3.5-Turbo, GPT-4o-mini, Llama-3.2-1B, Llama-3.2-3B, Mistral-7B)
Encoder-Based Models (ModernBert, RoBERTa)
Section 3.7
Approach 1: Prompt-Based Training (T5, GPT-3.5-Turbo, GPT-4o-min, Llama-3.2-1B, Llama-3.2-3B, Mistral-7B)
Approach 2: Fine-Tuned Model Training (T5, Llama-3.2-1B, Llama-3.2-3B, Mistral-7B, ModernBERT, RoBERTa)
Encoder-based models are not inherently designed for prompt-based learning
Models GPT-3.5-Turbo, GPT-4o-mini do not support fine-tuning.
Section 5.2 & 5.3
Evaluation Metrics (Accuracy, Recall, Precision, F1 Score
Section 5.4
Comparative Analysis

Fig. 3.1: Methodological Flow of Research

3.2 Dataset Description

The Microsoft Research Paraphrase Corpus (MRPC) is a widely used benchmark dataset for paraphrase identification tasks. It consists of 5801 pairs of sentences collected from online news sources. Each sentence pair is manually annotated as either a paraphrase (1) or non-paraphrase

(0). Specifically, 3900 pairs (67%) are paraphrases, while 1901 pairs (33%) are non-paraphrases. The dataset is already split into training (63%), validation (7%), and test (30%) sets. Since MRPC is part of the General Language Understanding Evaluation (GLUE) benchmark, it has been widely used in transformer-based NLP research, including models like BERT, T5, and RoBERTa. The dataset provides real-world sentence variations, capturing different ways the same information can be expressed across multiple sources, making it highly relevant for paraphrase detection tasks.

3.2.1 Data Selection

MRPC was selected for this study due to its high-quality human-labeled annotations, which ensure reliable evaluation. Unlike automatically generated paraphrase datasets, MRPC features expert-verified annotations with over 83% agreement among human annotators, making it a strong benchmark for evaluating model performance. Another key reason for selecting MRPC is its relatively small size (5801 sentence pairs). This allows for efficient fine-tuning and training without requiring extensive computational resources. Given the focus of this study on optimizing resource usage, MRPC provides an ideal dataset to analyze prompt-based vs. fine-tuned LLMs for paraphrase detection.

3.3 Data Pre-processing

Text preprocessing is a crucial step in NLP because raw text data often contains noise, inconsistencies, and redundant information that can negatively impact model performance. In paraphrase identification tasks, where models must determine semantic similarity between sentences, standardizing text formats and reducing irrelevant variations help improve accuracy and efficiency. By removing unnecessary characters, normalizing text, and structuring the data appropriately, we can enhance model interpretability, reduce computational complexity, and improve classification performance. For this research, I apply limited text preprocessing techniques to the MRPC dataset to ensure that sentence pairs are clean, structured, and meaningful for fine-tuned LLM models. Since prompt-based models work with raw text inputs and leverage contextual understanding, no preprocessing is applied for them. Additionally, because the MRPC dataset is already well-structured and human-annotated, only minimal preprocessing is necessary to maintain the integrity of the text while improving model efficiency. The limited preprocessing steps applied.

Preprocessing Steps for Noise Removal

- Remove special characters (except for punctuation essential for meaning).
- Remove extra spaces to ensure consistency in text input.
- Eliminate redundant dots (...) using regex-based filtering.
- Removing URLs and unnecessary symbols to reduce noise.
- Normalizing contractions (e.g., "don't" to "do not") to maintain textual clarity.
- Convert text to lowercase for standardization.

3.4 Data Transformation

The first step in data transformation is tokenization and encoding, where each sentence pair is broken into subwords. These tokenized words are then mapped to corresponding numerical representations using a pre-trained vocabulary. This step ensures that words with similar meanings are assigned similar embeddings, improving the model's ability to recognize paraphrases. Additionally, special tokens such as [CLS] (classification token) and [SEP] (separator token) are inserted between sentence pairs to provide structural context. Next, padding and truncation are applied to standardize input length. Since sentences in MRPC vary in length, all sequences must be transformed to a fixed size. Sentences shorter than the maximum length are padded with a special [PAD] token, while longer ones are truncated to prevent excessive memory usage. This ensures that the model processes batches of data efficiently without variation in sequence length. MRPC already provides binary labels, so one-hot encoding is not required.

3.5 Feature Engineering

To ensure high-quality inputs for model training, several feature engineering steps will be applied to the MRPC dataset. First, a missing value check will be conducted to confirm data completeness. Since MRPC is a well-structured dataset, missing values are unlikely, but validation will be performed to ensure no data loss during preprocessing. Next, outlier detection will be carried out, based on sentence length distributions to identify anomalies that may impact model performance.

3.6 Pre-Trained LLM Models

The selection of models for this study was driven by the need to evaluate a diverse range of state-of-the-art architectures, including decoder-based, encoder-based, and encoder-decoder-based models, to ensure comprehensive coverage of modern NLP capabilities. The chosen models GPT-3.5-Turbo, GPT-4o-mini, Llama-3.2-1B, Llama-3.2-3B, Mistral-7B, ModernBERT, and T5 represent the latest advancements in the field as of 2024 and were selected based on their unique strengths, efficiency, and applicability to the tasks under investigation.

Decoder-based Models:

- GPT-3.5-Turbo are among the most advanced and widely used decoder-based models developed by OpenAI. These models are particularly effective for prompt-based learning and few-shot learning, making them ideal for tasks that require high-quality text generation and understanding without extensive fine-tuning.
- GPT-4o-mini, a smaller variant of GPT-4, was chosen for its efficiency and cost-effectiveness compared to larger models. Despite its reduced size, it maintains competitive performance, making it suitable for applications where computational resources are limited.
- The Llama-3.2-1B and Llama-3.2-3B models, developed by Meta, are part of the latest generation of open-source decoder-based models. These models are designed to be lightweight and efficient, making them highly effective for tasks that require low computational overhead without sacrificing performance.
- Mistral-7B is a state-of-the-art open-source decoder-based model known for its high performance and scalability. It was included in this study to represent the capabilities of larger models and to provide a benchmark for comparing the efficiency of smaller models like Llama-3.2-1B and Llama-3.2-3B.

Encoder-based Models:

- ModernBERT are encoder-based models that excel in classification tasks and understanding contextual relationships in text. These models were chosen for their proven effectiveness in tasks such as paraphrase detection, sentiment analysis, and natural language understanding. Unlike decoder-based models, encoder-based models like ModernBERT are not designed for prompt-based learning but are highly effective when fine-tuned for specific tasks. Their inclusion in this study allows for a comparison between fine-tuning and prompt-based approaches.

Encoder-Decoder-based Models:

- T5 is an encoder-decoder-based model that treats all NLP tasks as a text-to-text problem, making it highly versatile and adaptable to a wide range of tasks, including paraphrase generation, summarization, and translation. T5 was selected for its ability to handle both generation and understanding tasks within a unified framework. Its flexibility and strong performance across diverse NLP tasks make it a valuable addition to this study.

3.6.1 General Transformers Architecture

All LLM models are based on Transformers. So, I will first discuss the Transformer architecture, explaining its key components and how it enables language modeling. Then, I will introduce the LLM models used in this research, detailing their structure, training methodology, and relevance to paraphrase identification.

Transformer is a type of neural network architecture. Transformer was developed by Google in 2017 (Vaswani et al., 2017). It is mainly developed to solve the problem of sequence transduction, or neural machine translation and some other tasks. It will learn the context of sequential data and generate new data out of it. Transformer's architecture has two main blocks: Encoder and Decoder.

Encoder: An encoder is used for processing the input sequence. It has a Multi-Head Attention mechanism and a fully connected Feed-Forward network, plus layer normalization for the output of each sub-layer. There are also residual connections around the two sub-layers.

Decoder: A decoder is used for generating the output sequence. It follows a similar structure, other than attention layers. It includes two types of attention sub-layers: masked multi-head attention and encoder-decoder attention.

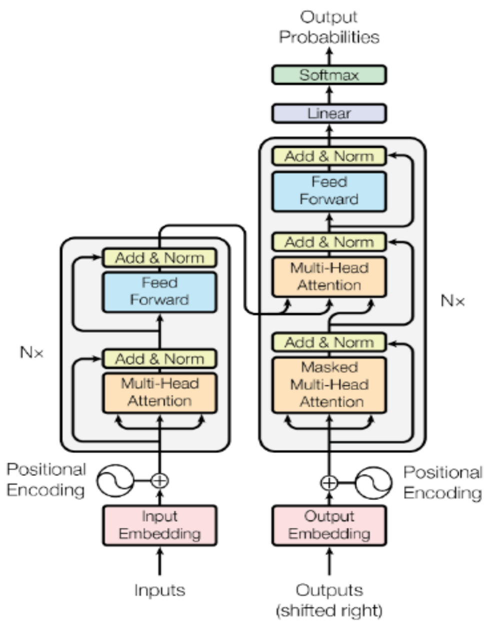


Fig. 3.2: The Transformer – Model Architecture (Vaswani et al., 2017)

Encoder each layer step by step:

Input Embedding: Input embedding is the first step of both encoder and decoder. It is responsible for converting the input text into numerical vectors of d_model dimensions. To prevent input embeddings becoming extremely small by normalizing them, multiplying them by the $\sqrt{d\ model}$.

Positional Encoding: Positional encoding is responsible for preserving sequence order and maintaining contextual information. This will help models to understand the sequential nature of data (like sentences). For positional encoding, they use sine and cosine functions with different frequencies for positional encoding.:

$PE_{(pos,2i)} = \sin(pos/10000^{2i/dmodel})$

$PE_{(pos,2i+1)} = \cos(pos/10000^{2i/dmodel})$

Where:

Pos = Position in sequence

i = Dimension of embedding (split into even and odd indices)

d = Dimension of the encoding

Add & Norm (Additional and Normalization): When we look at the encoder and decoder blocks, we see several normalization layers called add & norm. These layers are crucial for stabilizing and enhancing the learning process. It consists of two essential components: a residual connection (Add) and layer normalization (norm).

Residual Connection (add): Each sub-layer, including the self-attention and Feed Forward blocks, adds its output to its input before passing it to the Add & Norm layer. This process is known as skip connection. This approach allows the model to keep the original information from previous layers, helping gradients flow more smoothly and mitigating the vanishing gradient problem.
Normalization: Layer normalization formula is used. This process results in a normalized output with a mean of 0 and a standard deviation of 1.

$$\text{Norm}(X) = \frac{x-\mu}{\sigma+\varepsilon}$$

Where:

x: input data

μ: mean of the features

σ: Standard deviation

ε : epsilon (to avoid any divisions by zero)

Attention: Attention is the most crucial component of the Transformers. It is responsible for helping the model to understand complex relationship and patterns in the sentence. It assigns weights to each word based on the relevance and influence, works by seeing how similar and important each word is to all of the words in a sentence, including itself. This enables the model to capture long-range dependencies.

Self-Attention Mechanism: Self-attention, also known as scaled dot-product attention. The self-attention mechanism creates three vectors: Query (Q), Key (K), and Value (V). These vectors are learned during training.

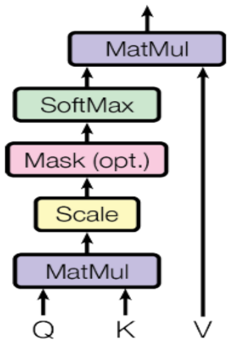


Fig. 3.3: Scaled Dot-Product Attention (Vaswani et al., 2017)

Attention Function:

$$\text{Attention}(Q,K,V) = \text{Softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$

Query (Q): Represents the word we are focusing on to calculate attention scores. Key (K): Represents all the other words in the sentence for comparison. Value (V): Represents the information from all the words, which will be used to create a weighted sum.

Attention Scores: The model finds attention scores by multiplying the Query vector of the current word with the Key vectors of all the words in the input. These scores indicate how related each word is to the current word.

Scale: The dot products are reduced by dividing them by the square root of the size of the key vectors.

SoftMax: The scaled dot products go through a SoftMax function to turn them into a probability distribution. This function makes sure the scores are between 0 and 1 and add up to 1.
Weighted Sum: The attention weights from SoftMax are used to create a weighted sum of the value vectors of all words. This sum represents the new version of the current word, including information from other words based on how important they are.

Multi-Head Attention: Instead of using just one attention mechanism, the model uses multiple heads, each with its own set of Query, Key, and Value vectors. This allows each head to focus on different parts of the input, capturing various aspects of word relationships. As per paper, they use 8 heads or parallel attention layers.

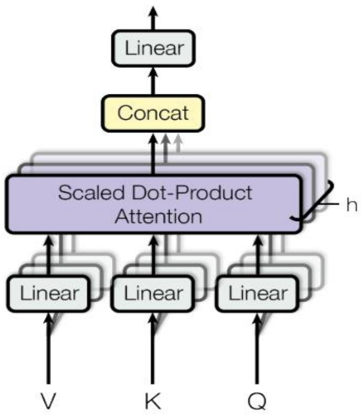


Fig. 3.4: Multi-Head Attention (Vaswani et al., 2017)

$$\text{MultiHead}(Q,K,V) = \text{concat}(\text{head}_1, \ldots.\text{head}_h)W^0$$

Where:

$\text{head}_i = \text{Attention}\ (QW_i^Q, KW_i^K, VW_i^V\ )$

$W_i^Q \in \mathbb{R}^{d_{model} \times d_k}$

$W_i^K \in \mathbb{R}^{d_{model} \times d_k}$

$W_i^V \in \mathbb{R}^{d_{model} \times d_v}$

$W_i^0 \in \mathbb{R}^{hd_v \times d_{model}}$

Feed-Forward: This is a fully connected feed-forward network layer. They applied two linear transformations with a ReLU activation in between. The dimensionality of input and output is $d_{model} = 512$, neurons (d_{ff}) are 2048.

$$\text{FFN}(x) = \max(0, xW_1 + b_1)W_2 + b_2$$

Where:

W_1 and W_2 are the weights.

b_1 and b_2 are the biases.

Decoder each layer:
As noted above, the decoder follows the same structure as the encoder, except for the attention layers.

Masked Multi-Head Self-Attention: This type of self-attention hides certain tokens in the input sequence or future tokens. This ensures that the decoder can only use information from past tokens and the current token to predict the next word in the output sequence.

Masked Attention Function:

$$\text{Masked Attention(Q,K,V)} = \text{Softmax}\left(\frac{QK^T}{\sqrt{d_k}} + mask\right)V$$

Where:

mask: large negative number (e.g., -inf)

Encoder-Decoder Attention: This layer allows the decoder to focus on important parts of the encoder's output. The decoder's current state generates the queries, while the keys and values come from the encoder's output. This helps the decoder generate accurate and contextually relevant responses.

3.6.2 Decoder-Based Models

3.6.2.1 Llama Architecture (Llama Team, 2024)

Llama is a family of transformer-based language models developed by Meta-AI. This is a family of autoregressive large language models. It is designed to be an efficient, open-source alternative to large-scale commercial models like OpenAI's GPT series. Llama models are particularly optimized for efficiency, making them suitable for research and fine-tuning on modest hardware compared to larger, proprietary models.
Llama follows the decoder-only transformer architecture, similar to GPT models, meaning it generates text autoregressively by predicting the next token in a sequence. It incorporates several optimizations to enhance efficiency and performance. Llama builds upon and significantly modifies the plain transformer.

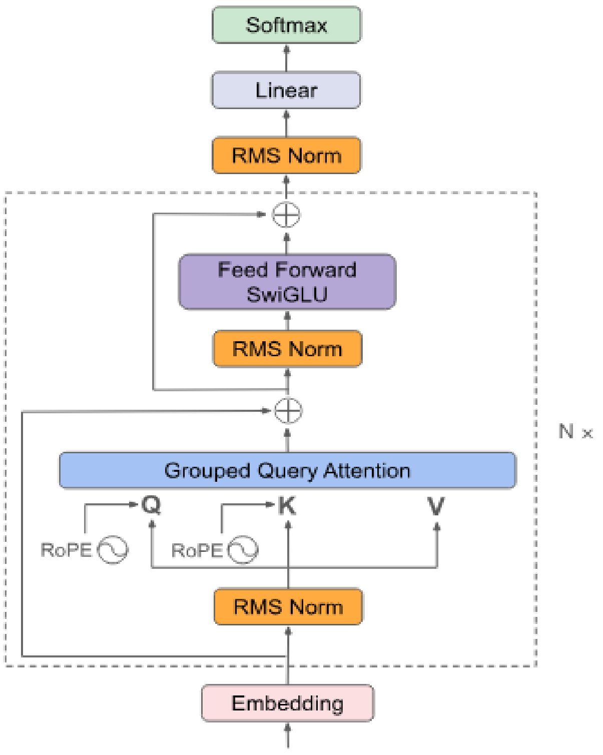


Fig. 3.5: Llama Architecture (Source: docs.nvidia.com)

RMS Norm: Instead of applying standard Layer Normalization, Llama uses RMSNorm (Root Mean Square Normalization) and applies it before each sub-layer. Unlike LayerNorm, which normalizes activations using mean and variance, RMS Norm normalizes using only the root mean square of the input activations. This reduces computational overhead while stabilizing training and improving performance in deep neural networks.

Rotary Positional Encoding (RoPE): RoPE replaces traditional absolute positional encodings by encoding relative positional information through rotational transformations. This allows the model to capture long-range dependencies more effectively. Unlike fixed positional embeddings, RoPE enables better generalization to longer sequences and helps improve the coherence of generated text.

Group Query Attention (GQA): GQA optimizes the self-attention mechanism by reducing the number of key-value pairs processed. Instead of using independent key-value pairs for every attention head, multiple query heads share the same key-value representations. This significantly reduces memory consumption and improves inference speed while maintaining the performance of full self-attention.

KV Cache (Key-Value Cache): KV Cache is a memory-efficient optimization used in autoregressive text generation. Instead of recomputing attention for every token in a sequence, the model stores and reuses previously computed key-value pairs. This reduces computational redundancy and speeds up inference, making it ideal for real-time applications.

SwiGLU Activation in Feedforward Networks : Llama replaces the standard ReLU activation with SwiGLU (Swish-Gated Linear Units) in its feedforward layers. SwiGLU introduces a gating mechanism where one branch applies the Swish activation while the other remains linear. This improves gradient flow, enhances expressiveness, and leads to better performance without a significant increase in computational cost.
Formula for SwiGLU:

$$\text{SwiGLU}(x) = (\text{Swish}(x) \,.\, W_1 x) W_2$$

Where: $\text{Swish}(x) = x * \text{sigmoid}(x)$

Single Layer Transformer Block Architecture: Each transformer block in Llama consists of multi-head self-attention, RMS Norm for stability, a feedforward network with SwiGLU activation, and residual connections. By optimizing these components, Llama achieves high efficiency while maintaining strong language modeling capabilities.

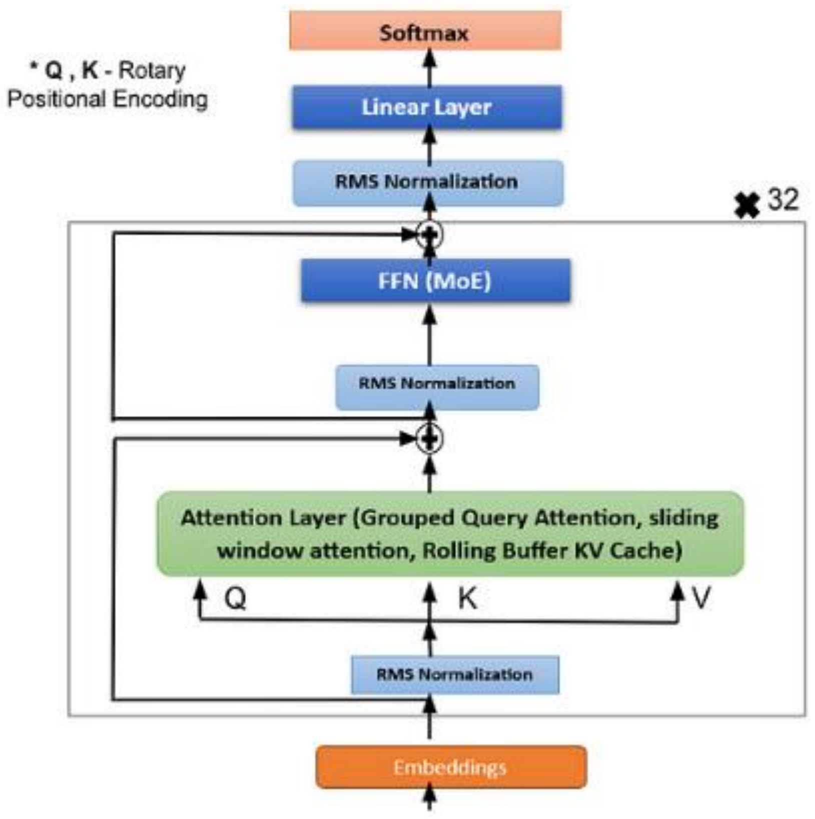
Popular Llama Model Variants & Parameter Sizes:

Table 3.1: Llama Model Variants

Name	Release date	Parameters	Context length (tokens)	Corpus size (tokens)
Llama	February 24, 2023	6.7B, 13B, 32.5B, 65.2B	2048	1–1.4T
Llama 2	July 18, 2023	6.7B, 13B, 69B	4096	2.2T
Llama 3	April 18, 2024	8B, 70.6B	8192	15T
Llama 3.1	July 23, 2024	8B, 70.6B, 405B	128000	15T
Llama 3.2	September 25, 2024	1B, 3B, 11B, 90B	128000	

Name	Release date	Parameters	Context length (tokens)	Corpus size (tokens)
Llama 3.3	December 7, 2024	70B	128000	

3.6.2.2 **Mistral Architecture** (Jiang et al., 2023a)



* Q , K - Rotary Positional Encoding
Softmax
Linear Layer
RMS Normalization
✖ 32
FFN (MoE)
RMS Normalization
Attention Layer (Grouped Query Attention, sliding window attention, Rolling Buffer KV Cache)
Q
K
V
RMS Normalization
Embeddings

Fig. 3.6: Mistral Architecture (Source towardsai.net)

Mistral, while building upon Llama, introduces its own set of advancements to push the envelope further:

Sliding Window Attention: This new attention mechanism allows Mistral to focus on subsets of the input data more efficiently, improving the model's ability to manage and interpret large sequences with greater precision.

Rolling Buffer KV Cache: An enhancement over Llama's KV cache, the rolling buffer mechanism optimizes memory usage further, enabling even more efficient handling of long sequences.

Mixture of Experts (MoE) in Feedforward Networks: Building on the SwiGLU-based feedforward of Llama, Mistral incorporates MoE to dynamically route different parts of the input through different specialized 'expert' networks, significantly boosting the model's capacity and efficiency.

Popular Mistral Model Variants & Parameter Sizes:

Table 3.2: Mistral Model Variants

Name	Release date	Parameters (Billion)	Context length (tokens)	Corpus size (tokens)
Mistral 7B	September 27, 2023	7.3B	32k	Not disclosed
Mistral Medium	November 25, 2024	13B	32k	Not disclosed
Mistral Small	November 25, 2024	22B	32k	220 B
Mistral Large	November 25, 2024	124 B	131k	Not disclosed

3.6.2.3 **GPT Architecture** (Yenduri et al., 2023)

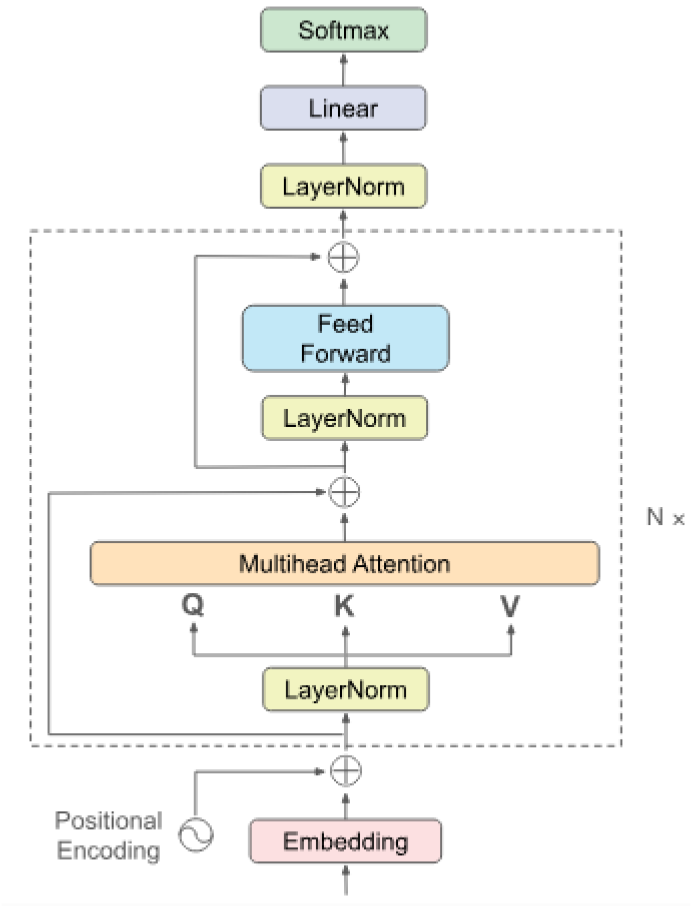


Fig. 3.7: GPT Architecture (Source docs.nvidia.com)

The GPT is a type of decoder-based transformer model designed primarily for text generation and language understanding tasks. It follows a left-to-right autoregressive architecture, where each word is predicted sequentially based on previous words in the text. Smaller versions of these models, like the GPT-3.5-turbo and GPT-4o Mini, are designed to be more efficient while still doing a great job.

Causal Self-Attention: Unlike bidirectional transformers (e.g., BERT), GPT uses causal self-attention, where each token can only attend to previous tokens in the sequence. This ensures that the model generates text in a left-to-right manner, making it suitable for autoregressive tasks.

Positionally Aware Tokenization: GPT employs absolute positional encodings added to word embeddings, ensuring that the model can distinguish between different positions in a sequence.

In later versions like GPT-4, more efficient positional encodings are used to improve long-context understanding.

LayerNorm Pre-Normalization: Instead of applying LayerNorm after each sub-layer, GPT models use pre-normalization before attention and feedforward layers. This stabilizes training and speeds up convergence.

Dense Feedforward Networks with GELU Activation: The model uses feedforward layers activated by GELU (Gaussian Error Linear Units), which improves non-linearity and enhances performance compared to traditional ReLU activations.

Popular OpenAI Model Variants & Parameter Sizes:

Table 3.3: GPT Model Variants

Name	Release date	Parameters	Context length (tokens)	Corpus size (tokens)
GPT-1	June 2018	117 million	-	-
GPT-2	Feb 2019	1.5 B	-	-
GPT-3	June 2020	175B	2048	570B
GPT-3.5	March 2022	175B	4096	570B
GPT-4	March 2023	Not disclosed	8192 - 32768	Not disclosed
GPT-4o	May 2024	Not disclosed	Not disclosed	Not disclosed
GPT-4o Mini	July 2024	Not disclosed	Not disclosed	Not disclosed

3.6.2.4 Comparison Table

Table 3.4: Comparison Table: Standard Transformer vs. GPT vs. Llama vs. Mistral

Feature	Standard Transformer (Original)	GPT Model (OpenAI)	Llama Model (Meta AI)	Mistral Model (Mistral AI)
Architecture	Encoder-Decoder (Seq2Seq)	Decoder-Only	Decoder-Only	Decoder-Only
Core Use Case	Machine Translation, Summarization	Text Generation, Chatbots	Open-source LLM for Research	High-performance Open LLM
Training Paradigm	Pretrained + Fine-Tuned	Autoregressive (Next-token prediction)	Pretrained + Fine-Tuned	Pretrained + Fine-Tuned

Model Type	BERT, T5, etc.	GPT-3.5, GPT-4, GPT-4o	Llama 2, Llama 3	Mistral 7B
Token Processing	Bi-directional (Enc-Dec)	Left-to-Right	Left-to-Right	Left-to-Right
Context Length	512-2048 tokens	16K-128K tokens	4K-128K tokens	8K-32K tokens
Multimodal Support	No	Yes (GPT-4o)	Yes (Llama 3.2)	No (Mistral-7B)
Open-Source?	No (Mostly Proprietary)	No	Yes	Yes
Fine-Tuning	Required for Customization	Limited API Access	Fully Tunable	Fully Tunable
Efficiency	Heavy Computation	Optimized (GPT-4o Mini)	Highly Optimized	Highly Optimized (Mixture of Experts)
Inference Cost	High	High (GPT-4), Lower (GPT-4o Mini)	Lower than GPT	Lower than Llama
Example Use Case	Google Translate, BERT QA	ChatGPT, Copilot, DALL·E	Academic Research, AI Models	Fast, Lightweight AI models

3.6.3 Encoder-Based Models

BERT: BERT is a groundbreaking language model that leverages the transformer architecture's encoder component to learn deep, bidirectional representations of text. It's designed to understand the context of words within a sentence by processing the entire sequence simultaneously, rather than sequentially. BERT's pre-training involves two main tasks: Masked Language Modeling (MLM), where random words are masked and the model predicts them, and Next Sentence Prediction (NSP), where the model determines if two sentences follow each other. This pre-training enables BERT to generate highly contextualized word embeddings, which can then be fine-tuned for various downstream NLP tasks.

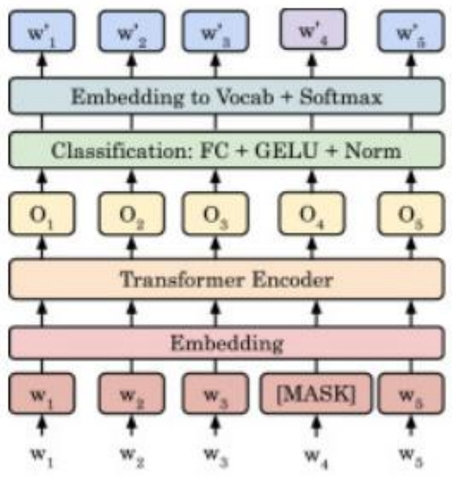


Fig. 3.8: BERT Architecture

Key Points of BERT:

- Transformer Encoder: Uses the encoder part of the Transformer architecture.
- Bidirectional Processing: Processes input sequences from both directions.
- Masked Language Modeling (MLM): Predicts masked words in a sentence.
- Next Sentence Prediction (NSP): Determines if two sentences follow each other.
- Contextualized Word Embeddings: Generates word representations based on surrounding context.

3.6.3.1 **ModernBERT** (Warner et al., 2024)

ModernBERT is an advanced variant of the BERT model, designed to improve efficiency, generalization, and contextual understanding while maintaining strong performance across NLP tasks. Like BERT, it is based on a multi-layer bidirectional transformer encoder that captures deep contextual relationships between words. However, ModernBERT introduces architectural optimizations that reduce computational costs without significantly compromising accuracy. These optimizations make it particularly suitable for large-scale applications such as paraphrase identification, question answering, and text classification.

One of the key enhancements in ModernBERT is efficient attention mechanisms, which replace BERT's full self-attention with sparse or approximate attention methods. This reduces the quadratic complexity associated with self-attention while preserving the ability to model long-range dependencies in text. Additionally, layer pruning and parameter reduction techniques streamline the architecture, making ModernBERT faster and more memory-efficient. Some versions of ModernBERT also utilize knowledge distillation, where a smaller model learns from a larger, more complex model, ensuring efficient performance without excessive resource usage.

ModernBERT improves upon BERT's tokenization strategy by refining dynamic positional embeddings and adaptive vocabulary handling to better process out-of-vocabulary words. These enhancements contribute to superior generalization across different text domains. Moreover, the model supports task-specific fine-tuning, where dropout rates, learning rates, and layer freezing techniques are adjusted based on the dataset and application. Such optimizations make ModernBERT a versatile and efficient alternative to traditional BERT models, striking a balance between computational efficiency and language understanding.

Difference Between BERT and ModernBERT:

Context Length: ModernBERT supports significantly longer context windows than BERT.

Architecture Enhancements: ModernBERT incorporates architectural improvements like RoPE, GeGLU, and optimized attention mechanisms (Flash attention, and alternating global and local attention).

Training Data: ModernBERT is trained on a more diverse dataset, including code.

Efficiency: ModernBERT is optimized for speed and memory efficiency on modern hardware.

Activation Functions: ModernBERT uses more advanced activation functions like GeGLU.

3.6.3.2 Comparison Table

Table 3.5: Comparison Table: Standard BERT vs. ModernBERT

Feature	BERT	ModernBERT
Core Architecture	Transformer Encoder	Enhanced Transformer Encoder
Training Data	BooksCorpus, English Wikipedia	Expansive & diverse, including code, and modern text sources.
Masking	Static masking	Dynamic Masking

NSP	Used	Removed
Training Procedure	Standard training	Optimized for efficiency, & long context handling.
Vocabulary	Standard BERT vocabulary	larger than BERT
Context Length	512 tokens	8192 tokens
Positional Embeddings	Traditional Positional Embeddings	RoPE
Activation Functions	ReLU	GeGLU
Attention mechanism	Standard Multi-Head Attention	Flash attention, and alternating Global and local attention patterns.

3.6.4 Encoder-decoder-based Model

3.6.4.1 T5 (Text-to-Text Transfer Transformer)

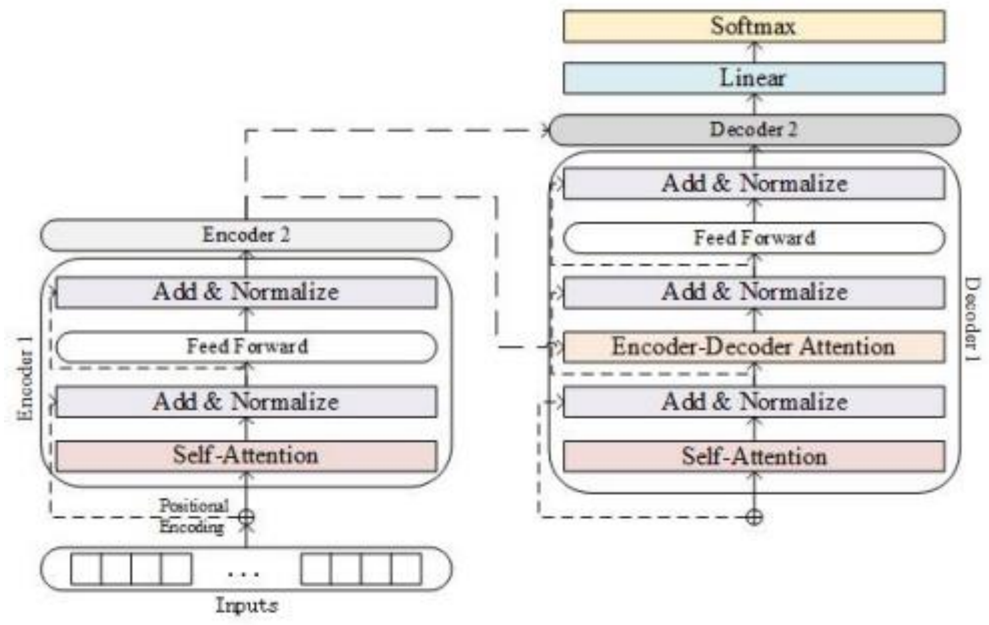


Fig. 3.9: T5 Architecture (Wang et al., 2023)

The T5 model by Google is a sequence-to-sequence (encoder-decoder) transformer designed to handle various NLP tasks in a unified text-to-text format. Unlike traditional models that require task-specific architectures, T5 converts every NLP task into a text generation problem, making it highly flexible. It consists of an encoder that processes input text and a decoder that generates the output, using self-attention and cross-attention mechanisms to understand context. Instead of the standard MLM approach, T5 is pretrained using span corruption, where random text spans are removed, and the model learns to reconstruct them, improving its understanding of sentence structures.

3.7 Model Training Approach

In this study, I propose a two-part approach to identify paraphrases using prompt-based learning and fine-tuned LLMs. LLMs offer various ways to tackle this problem, such as prompt-based methods that use pre-trained models without extra training, and fine-tuned models that are adapted using specific datasets to improve performance. This research compares these methods in terms of accuracy, efficiency, and computational cost, focusing on decoder-based models, encoder-based models and encoder-decoder-based models.

Approach Overview

- Prompt-based LLMs - Utilizing pre-trained models with structured prompts to extract paraphrase similarity, requiring no additional training.
- Fine-tuned LLMs - These models are trained on a labeled paraphrase dataset using LoRA & QLoRA fine-tuning to enhance performance.

Both methods are evaluated using standard NLP metrics like accuracy, precision, recall, and F1-score, while also considering computational cost and efficiency.

3.7.1 Prompt-Based Training Approach

3.7.1.1 Prompt Engineering

Prompt engineering is the process of designing structured input instructions that guide a generative AI model to produce desired outputs. In the context of paraphrase identification, effective prompts help models understand semantic similarities between sentence pairs without requiring explicit fine-tuning. However, crafting optimal prompts can be challenging, as even small changes in the input text such as rephrasing a sentence, altering word order, or modifying punctuation can significantly impact the model's response and prediction accuracy.

Role of In-Context learning (ICL)

In-context learning (ICL) is a powerful technique where examples or additional context are embedded directly within the prompt to improve the model's performance on specific tasks. For paraphrase detection, ICL allows the model to recognize sentence equivalence based on provided examples, enhancing its ability to generalize across different paraphrasing styles. This technique is particularly useful for zero-shot and few-shot learning, where explicit model fine-tuning is not performed.

- Zero-Shot Learning: The model performs paraphrase identification without any task-specific examples, relying solely on its pre-trained knowledge. This is useful when labeled data is limited but may result in lower accuracy.
- Few-Shot Learning: A few demonstration examples (e.g., correctly labeled paraphrase pairs) are included in the prompt to guide the model’s behaviour. This helps refine its understanding and significantly improves performance compared to zero-shot prompting.

3.7.1.2 Models Used

The following pre-trained models were utilized for paraphrase identification:

- GPT-4o Mini & GPT-3.5 Turbo (OpenAI API) - Highly capable decoder-based models accessible via OpenAI API, offering strong language understanding and generation capabilities.
- Llama-3.2-1B, Llama-3.2-3B, Mistral-7B - Open-source decoder-based models known for their efficiency and adaptability across various NLP tasks.
- ModernBERT – Lightweight encoder-based models optimized for contextual understanding, providing robust performance in classification and text representation tasks.
- T5 – A lightweight encoder-decoder-based model designed for sequence-to-sequence tasks, excelling in text generation, summarization, and translation.

3.7.1.3 Prompt Design

A well-designed prompt ensures the model correctly interprets and classifies paraphrases. The prompt template is structured to guide the model in identifying paraphrases accurately.

Prompt Template:

Sample 1:

```
prompt="""
We have two sentences. Determine if they are paraphrases of each other.
A paraphrase conveys the same meaning using different words while maintaining the core information. A non-paraphrase has a different meaning or significantly alters the information.
Respond strictly with one word: **paraphrase** or **non-paraphrase**.
   Now classify:
   Sentence 1: "{sentence1}"
   Sentence 2: "{sentence2}"
   Answer:  """
```

3.7.2 Fine-Tuned Model Training

Fine-tuning is a critical step in adapting pre-trained LLMs for paraphrase identification, ensuring they learn domain-specific patterns effectively. This process involves supervised fine-tuning on paraphrase sentence pairs, allowing the model to capture sentence-level semantic similarity with high accuracy. Learning rate tuning plays a crucial role in balancing model generalization and preventing overfitting, ensuring that the model does not memorize training data but instead learns meaningful representations. Additionally, batch size optimization is employed to improve training efficiency, ensuring stable convergence while making optimal use of available memory resources. These techniques collectively enhance the model's ability to differentiate between paraphrased and non-paraphrased sentences, improving overall performance in real-world NLP applications.

3.7.2.1 LLM Models Fine-tuning Techniques

Parameter Efficient Fine-Tuning (PEFT): PEFT is a transfer learning approach that tackles the challenges of fully fine-tuning LLMs. It involves keeping all the pre-trained model's original parameters unchanged and adding new parameters that can be adjusted during fine-tuning. Adapters are a type of submodule added to pre-trained models to adjust their hidden representations during fine-tuning. By placing adapters after the multi-head attention and feed-forward layers in the transformer architecture, only the adapter parameters are updated during fine-tuning, while the rest of the model remains frozen. There are different methods for PEFT, with LoRA and QLoRA being the most popular and effective. Benefits of PEFT: Decreased computational and storage costs, Overcoming catastrophic forgetting, portability, sustainability, lower storage requirements etc.

3.7.2.1.1 LoRA (Low-Rank Adaptation)

LoRA is a parameter-efficient fine-tuning technique primarily developed by Microsoft. It significantly reduces the number of trainable parameters by up to 10,000 times and lowers GPU memory requirements by 3 times, making it highly efficient for adapting large models with minimal computational cost. By introducing low-rank matrices into the weight update process, LoRA allows fine-tuning without modifying the original model weights, preserving pre-trained knowledge while enhancing adaptability. This approach enables researchers and developers to fine-tune large-scale models on resource-constrained hardware without compromising performance.

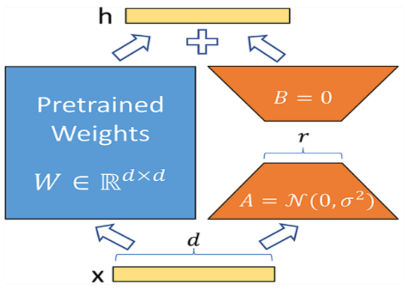

h
Pretrained Weights
$W \in \mathbb{R}^{d \times d}$
$B = 0$
r
$A = \mathcal{N}(0, \sigma^2)$
d
x

Fig. 3.10: LoRA Reparameterization (Hu et al., 2021)

LoRA decomposes the weight matrix(W_0) into the two smaller matrices A and B such that $\Delta W = BA$, where A and B have much lower ranks than W_0. During finetuning, only the matrices A and B are updated while W_0 remains fixed.

$h = W_0 + \Delta W = W0 + BA$

Where:

W_0: Actual pre-trained Weights

$W_0 \in \mathbb{R}^{d \times k}$

A & B: Two smaller matrix

$A \in \mathbb{R}^{d \times r}, B \in \mathbb{R}^{r \times k}$

r: hyperparameter (values1,2,4,8 or 64)

$r << \min(d, k)$

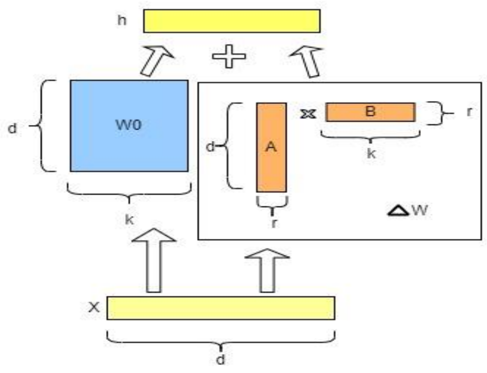


Fig. 3.11: LoRA Adaptation

3.7.2.1.2 QLoRA (Quantized Low-Rank Adaptation)

QLoRA is an extension of LoRA that makes the method even more efficient. Quantization is a process of reducing the precision of the numbers used to represent model weights. For instance, instead of storing weights as 32-bit floating points, you might store them using just 4 bits. This significant reduction in data size means that the model requires less memory, and computations can be executed more quickly and with less energy.

3.7.2.2 Models Used

The following pre-trained models were utilized for paraphrase identification:

- Llama-3.2-1B, Llama-3.2-3B, Mistral-7B - Open-source decoder-based models known for their efficiency and adaptability across various NLP tasks.
- ModernBERT– Lightweight encoder-based models optimized for contextual understanding, providing robust performance in classification and text representation tasks.
- T5 – A lightweight encoder-decoder-based model designed for sequence-to-sequence tasks, excelling in text generation, summarization, and translation.

3.7.2.3 Fine-Tuning Approach

Decoder-Based Models:

To compare with prompt-based methods, a fine-tuned version of Llama-3.2-1B, Llama-3.2-3B, Mistral-7B are developed using QLoRA and LoRA, a technique for efficient fine-tuning with minimal GPU memory usage. Llama-3.2-1B was chosen for its lightweight design and efficiency in paraphrase tasks.

Training Approach

- QLoRA fine-tuning to adapt pre-trained weights with minimal GPU memory usage.
- Learning rate tuning to prevent overfitting while optimizing convergence speed.
- Batch size optimization to balance memory consumption and efficiency.

Evaluation:

- Performance comparison on MRPC test set.
- Analyse computational efficiency compared to prompt-based models.

Encoder-Based & Encoder-Decoder-Based Approach

To ensure a comprehensive comparison, both encoder-based and encoder-decoder-based transformer models were fine-tuned for paraphrase identification. These models, such as ModernBERT and T5, transform input sentences into high-dimensional vector representations, capturing semantic relationships effectively. The fine-tuned models utilize classification heads to predict paraphrase relationships with improved accuracy and efficiency. Encoder-based models, like ModernBERT focus on contextualized embeddings for sentence understanding, while encoder-decoder models, such as T5, leverage a sequence-to-sequence structure to enhance text comprehension and transformation.

3.8 Evaluation Metrics

Train test split is not needed as data has already split. I will use standard binary classification metrics to evaluate our models:

Confusion Matrix: A confusion matrix displays the counts of correct and incorrect predictions made by a classification model, comparing them to the actual outcomes in the data.

Accuracy: Accuracy measures the total number of correct classifications divided by the total number of cases.

Recall/True Positive Rate/ Sensitivity: Recall/Sensitivity tells us out of the actual positive values, how many of them have predicted as positive.

Precision: Precision tells us out of all the predicted positive values, how many of them are actually positive.

Specificity: Specificity tells us out of all the actual negatives, how many of them are correctly predicted negative.

F1 Score: The F1 Score is a single metric that is a harmonic means of precision and recall.

ROC (Receiver Operating Characteristics) AUC (Area Under Curve) Score: It is a graphical representation of the model performance at various threshold levels. It shows the trade-off between the true-positive rate and false-positive rate.

3.9 Required Resource

Computational Resources

- Single-GPU Instances: Used for small to medium-scale models like Llama-3.2-1B, and ModernBERT.
- Multi-GPU or High-Memory GPUs: Required for fine-tuning large models like Mistral-7B.
- Cloud Service Providers: AWS, JarvisLabs, on-premise HPC clusters, or any cloud provider.

GPU Configrations (Used JarvisLabs.ai)

- Single NVIDIA RTX5000: 7 CPUs | 32GB RAM | 16GB VRAM | Quadro Gen | $0.46/hr + Storage Cost
- Single NVIDIA A5000: 32 CPUs | 64GB RAM | 24GB VRAM | Ampere Gen | $0.46/hr + Storage Cost
- Single NVIDIA A100: 7 CPUs | 32GB RAM | 40GB VRAM | Ampere Gen | $1.20/hour + Storage Cost

GPU Models Used

- NVIDIA A5000 / RTX5000: Used for smaller models.
- NVIDIA A100: Required for Mistral-7B due to its high memory needs.

GPU Memory Requirements

- Small models (~8GB): Llama-1B, ModernBERT.
- Medium models (~16GB): Llama-3B.
- Large models (~32GB+): Mistral-7B.
- Disk Storage: Required for datasets, fine-tuned checkpoints, and logs.

Software & Frameworks

- Deep Learning Libraries
  - Core Frameworks: PyTorch / TensorFlow, Hugging Face Transformers
  - Optimization & Training: datasets, accelerate, peft, bitsandbytes

    - Tokenizer & Evaluation: sentencepiece, evaluate, huggingface_hub
    - LLM Utilities: langchain, openai
- Machine Learning Libraries
    - Data Handling & Processing: pandas, numpy
    - Visualization: matplotlib, seaborn
    - ML Algorithms: sklearn, scipy, tdm, dotenv
- NLP Libraries
    - Text Processing: collections, re, nltk, itertools, spacy
    - Embedding & Sentence-Level Analysis: sentence_transformers
- Development Environment
    - Cloud based Jupyter Notebook and Visual Studio Code (VS Code).

3.10 Summary

This chapter explains the methods used to identify paraphrases using LLMs, focusing on two main approaches: prompt-based learning and fine-tuning. The research starts with selecting the MRPC, a well-known dataset with labeled sentence pairs. To prepare the data, preprocessing steps like text normalization, lowercasing, and removing unnecessary characters are applied. The data is also tokenized to make it compatible with transformer-based models. The study evaluates popular transformer models GPT-3.5-Turbo, GPT-4o-min, Llama-3.2-1B, Llama-3.2-3B, Mistral-7B, ModernBERT, and T5 to see how well they perform in both zero-shot learning (without additional training) and fine-tuned scenarios. Fine-tuning is done using LoRA and its more efficient version, QLoRA, which helps adapt the models to the task while saving computational resources.

In the prompt-based approach, structured prompts are designed to guide pre-trained models in identifying paraphrases. This method uses zero-shot and few-shot learning techniques, and experiments with temperature tuning to improve the model's consistency. On the other hand, fine-tuned models are trained on task-specific data, which helps them better understand and generalize to new paraphrase examples. The performance of both approaches is measured using standard NLP metrics like accuracy, precision, recall, and F1 score. This allows for a thorough comparison of the two methods, considering factors like accuracy, computational cost, and efficiency. The goal is to find the most effective and practical approach for paraphrase identification. In short, this chapter outlines a clear and systematic way to compare prompt-based, fine-tuned decoder-based LLMs, encoder-based LLMs and encoder-decoder-based LLMs helping to determine which method works best for identifying paraphrases.

CHAPTER 4

ANALYSIS & DESIGN

4.1 Introduction

The Analysis & Design phase is crucial for understanding the dataset, evaluating model performance, and selecting the best configurations for both prompt-based learning and fine-tuning approaches. This chapter provides a detailed examination of the MRPC dataset, focusing on its structure, preprocessing steps, and EDA. The study explores different prompt engineering techniques and hyperparameter tuning strategies across various model architectures, including GPT-3.5-Turbo, Llama-3.2-1B, Llama-3.2-3B, Mistral-7B, ModernBERT and T5. A key aspect of this analysis is the evaluation of model performance in terms of accuracy, computational efficiency, and cost, particularly when deploying models via APIs (such as OpenAI's GPT models) or downloading them from Hugging Face (HF) for fine-tuning. The chapter also presents fine-tuning experiments using LoRA and QLoRA approaches to optimize training configurations while addressing challenges related to hardware limitations and resource constraints. Additionally, the challenges faced during prompt engineering, model fine-tuning, and optimization strategies are discussed.

The chapter is structured as follows: Section 4.2 provides an in-depth analysis of the MRPC dataset, covering preprocessing steps, noise detection, and EDA. Section 4.3 discusses different prompt-based learning approaches, including prompt formulation, hyperparameter tuning. Section 4.4 focuses on fine-tuning analysis, detailing optimized training configurations, cost evaluations, and computational constraints. Finally, Section 4.6 presents a summary of key insights that lay the groundwork for further analysis in the next chapter, Results & Discussion.

4.2 Exploratory Data Analysis (EDA)

4.2.1 MRPC Dataset Overview

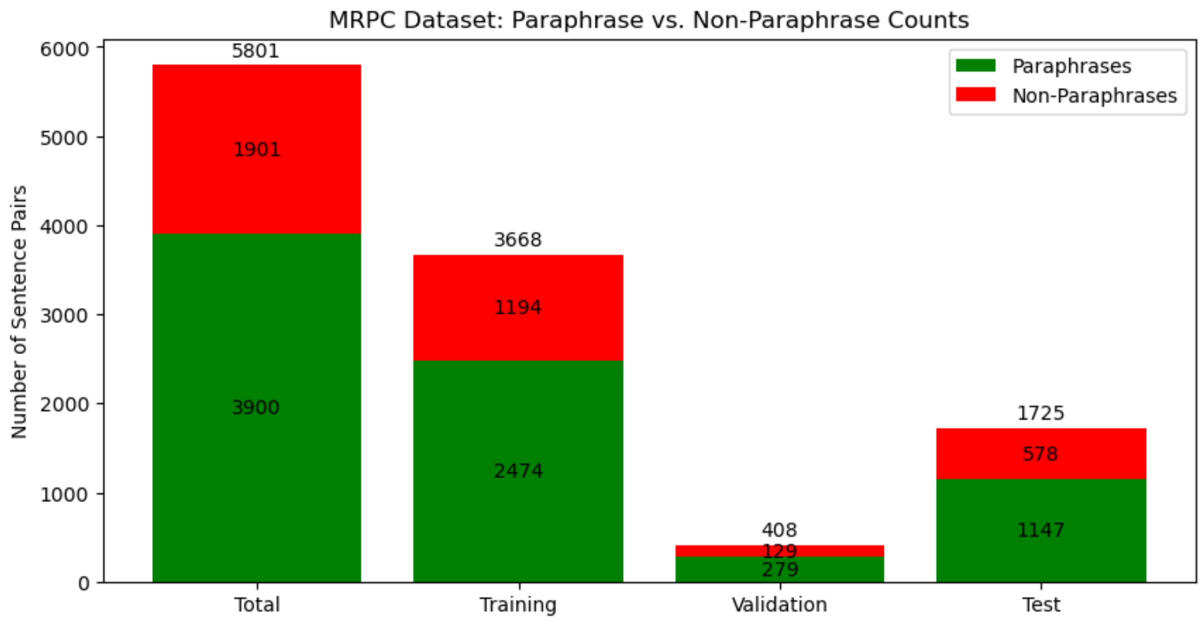


Fig. 4.1: MRPC Dataset Split: Paraphrase vs. Non-Paraphrase Counts

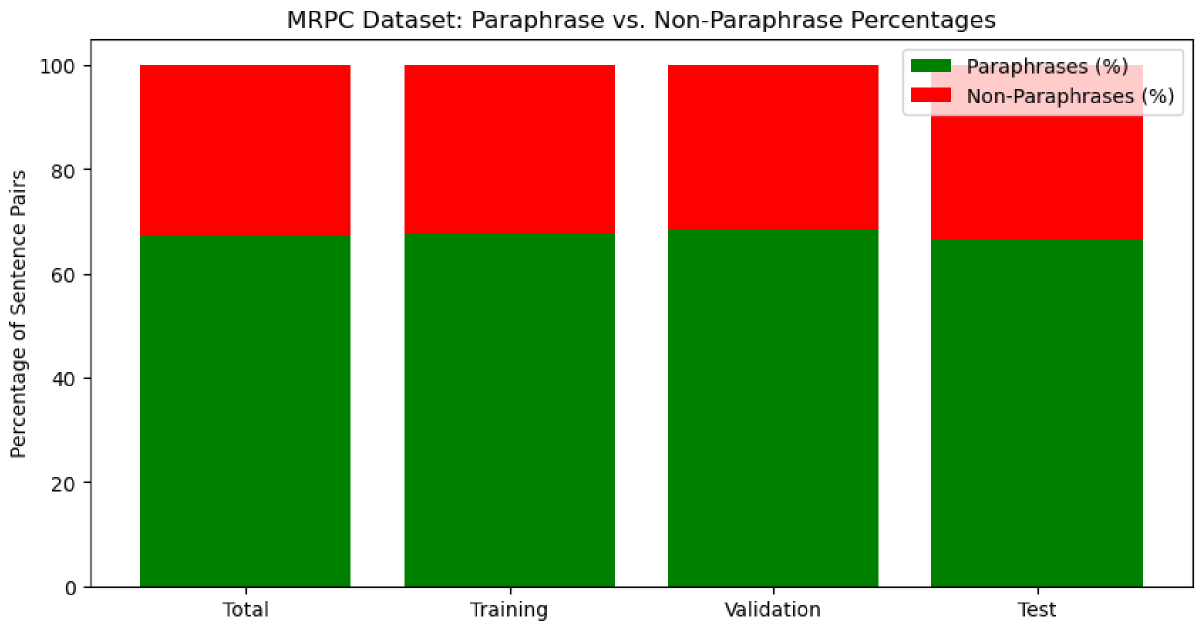


Fig. 4.2: MRPC Dataset Split: Paraphrase vs. Non-Paraphrase Percentages

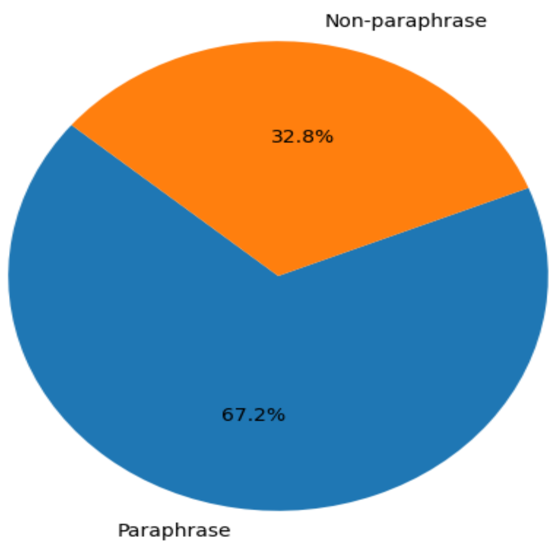


Fig. 4.3: Overall Paraphrase Distribution

Key Observations:

- The dataset exhibits a slight imbalance, with approximately twice as many paraphrase examples (around 67%) as non-paraphrase examples (around 33%) across all subsets. This imbalance is consistent across the training, validation, and test sets, suggesting a systematic characteristic of the dataset.
- The proportion of paraphrase to non-paraphrase examples is remarkably consistent across all three splits. Each split holds roughly 67-68% paraphrases, and 32-33% non-paraphrases. This consistency is positive for training and testing models, as it helps to ensure that models generalize well across different portions of the data.
- The split of the total dataset is heavily weighted toward the training data. This is common practice, as large training sets are needed to properly train machine learning models.
- The test set is a significant portion of the data, which allows for robust testing of model performance.
- The validation set is the smallest, as it is only used for hyperparameter tuning, and model evaluation during the training process.

4.2.2 Dataset Structure and Preprocessing

The MRPC dataset contains 4 columns:

Sentence1: The first sentence in the pair.

Sentence2: The second sentence in the pair.

Label: Indicates whether the pair is a paraphrase (1) or non-paraphrase (0).

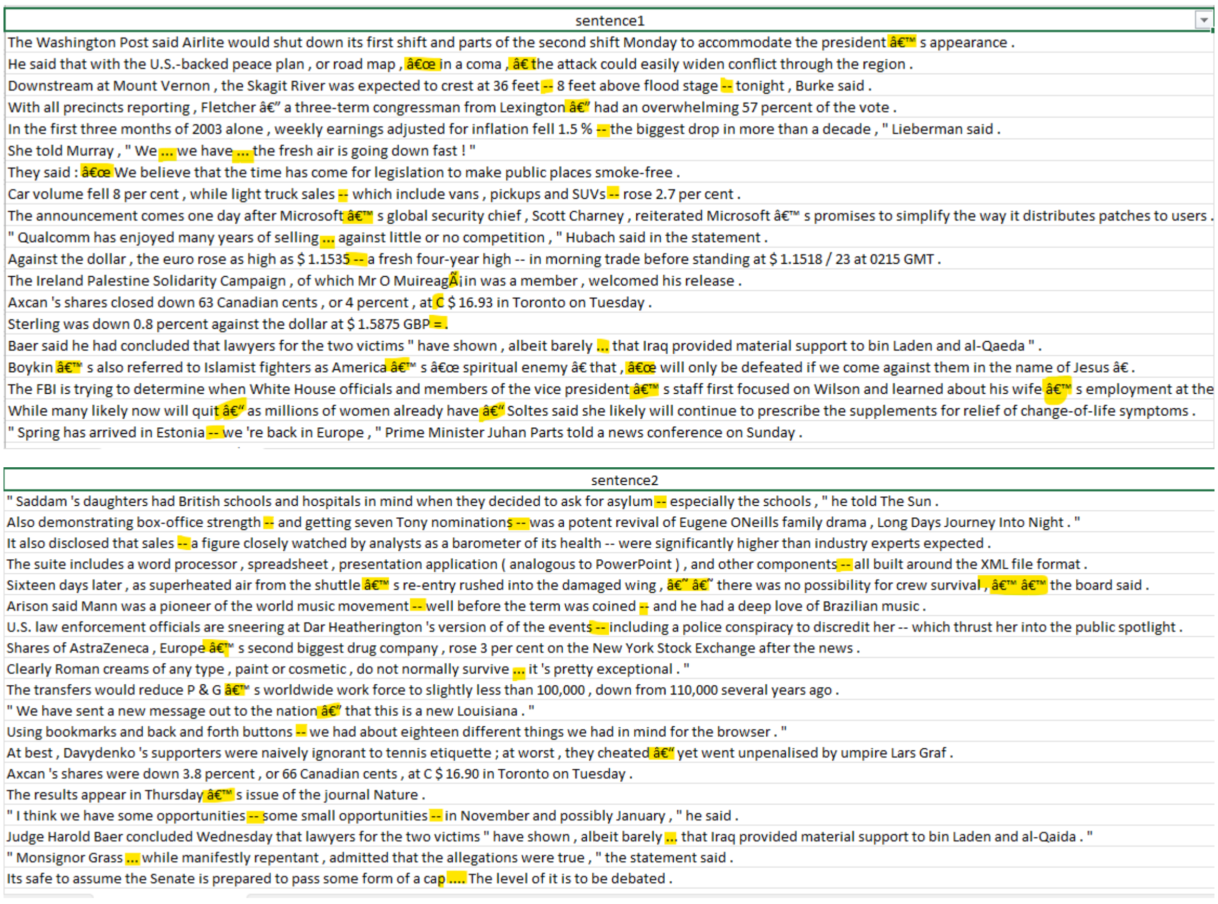
Idx: An index column, A unique identifier for each sentence pair.

During preprocessing, the idx column was dropped as it is not relevant for model training or evaluation. The dataset was checked for missing values, and none were found, ensuring the dataset is complete and ready for analysis. This ensures that no imputation techniques (such as replacing missing text with placeholders) are required.

4.2.3 Noise Detection and Removal

Despite having no missing values, the dataset contains various forms of noise, which can degrade model performance if not addressed. The noise includes:

4.2.3.1 Original Dataset Snapshot (Before Noise Removal)



sentence1
The Washington Post said Airlite would shut down its first shift and parts of the second shift Monday to accommodate the president â€™ s appearance .
He said that with the U.S.-backed peace plan , or road map , â€œ in a coma , â€ the attack could easily widen conflict through the region .
Downstream at Mount Vernon , the Skagit River was expected to crest at 36 feet -- 8 feet above flood stage -- tonight , Burke said .
With all precincts reporting , Fletcher â€" a three-term congressman from Lexington â€" had an overwhelming 57 percent of the vote .
In the first three months of 2003 alone , weekly earnings adjusted for inflation fell 1.5 % -- the biggest drop in more than a decade , " Lieberman said .
She told Murray , " We ... we have ... the fresh air is going down fast ! "
They said : â€œ We believe that the time has come for legislation to make public places smoke-free .
Car volume fell 8 per cent , while light truck sales -- which include vans , pickups and SUVs -- rose 2.7 per cent .
The announcement comes one day after Microsoft â€™ s global security chief , Scott Charney , reiterated Microsoft â€™ s promises to simplify the way it distributes patches to users .
" Qualcomm has enjoyed many years of selling ... against little or no competition , " Hubach said in the statement .
Against the dollar , the euro rose as high as $ 1.1535 -- a fresh four-year high -- in morning trade before standing at $ 1.1518 / 23 at 0215 GMT .
The Ireland Palestine Solidarity Campaign , of which Mr O MuireagÃ¡in was a member , welcomed his release .
Axcan 's shares closed down 63 Canadian cents , or 4 percent , at C $ 16.93 in Toronto on Tuesday .
Sterling was down 0.8 percent against the dollar at $ 1.5875 GBP = .
Baer said he had concluded that lawyers for the two victims " have shown , albeit barely ... that Iraq provided material support to bin Laden and al-Qaeda " .
Boykin â€™ s also referred to Islamist fighters as America â€™ s â€œ spiritual enemy â€ that , â€œ will only be defeated if we come against them in the name of Jesus â€ .
The FBI is trying to determine when White House officials and members of the vice president â€™ s staff first focused on Wilson and learned about his wife â€™ s employment at the
While many likely now will quit â€" as millions of women already have â€" Soltes said she likely will continue to prescribe the supplements for relief of change-of-life symptoms .
" Spring has arrived in Estonia -- we 're back in Europe , " Prime Minister Juhan Parts told a news conference on Sunday .

sentence2
" Saddam 's daughters had British schools and hospitals in mind when they decided to ask for asylum -- especially the schools , " he told The Sun .
Also demonstrating box-office strength -- and getting seven Tony nominations -- was a potent revival of Eugene ONeills family drama , Long Days Journey Into Night . "
It also disclosed that sales -- a figure closely watched by analysts as a barometer of its health -- were significantly higher than industry experts expected .
The suite includes a word processor , spreadsheet , presentation application (analogous to PowerPoint) , and other components -- all built around the XML file format .
Sixteen days later , as superheated air from the shuttle â€™ s re-entry rushed into the damaged wing , â€˜ â€˜ there was no possibility for crew survival , â€™ â€™ the board said .
Arison said Mann was a pioneer of the world music movement -- well before the term was coined -- and he had a deep love of Brazilian music .
U.S. law enforcement officials are sneering at Dar Heatherington 's version of of the events -- including a police conspiracy to discredit her -- which thrust her into the public spotlight .
Shares of AstraZeneca , Europe â€™ s second biggest drug company , rose 3 per cent on the New York Stock Exchange after the news .
Clearly Roman creams of any type , paint or cosmetic , do not normally survive ... it 's pretty exceptional . "
The transfers would reduce P & G â€™ s worldwide work force to slightly less than 100,000 , down from 110,000 several years ago .
" We have sent a new message out to the nation â€" that this is a new Louisiana . "
Using bookmarks and back and forth buttons -- we had about eighteen different things we had in mind for the browser . "
At best , Davydenko 's supporters were naively ignorant to tennis etiquette ; at worst , they cheated â€" yet went unpenalised by umpire Lars Graf .
Axcan 's shares were down 3.8 percent , or 66 Canadian cents , at C $ 16.90 in Toronto on Tuesday .
The results appear in Thursday â€™ s issue of the journal Nature .
" I think we have some opportunities -- some small opportunities -- in November and possibly January , " he said .
Judge Harold Baer concluded Wednesday that lawyers for the two victims " have shown , albeit barely ... that Iraq provided material support to bin Laden and al-Qaida . "
" Monsignor Grass ... while manifestly repentant , admitted that the allegations were true , " the statement said .
Its safe to assume the Senate is prepared to pass some form of a cap .... The level of it is to be debated .

Fig. 4.4: Snapshot of Dataset Before Noise Removal

Special Characters and Punctuation:

- Certain sentences contain special characters such as < .SPX >, â€™, Â½, Â£, < .IXIC, @, <, >, (,), which do not contribute to meaning.
- The dataset also has multiple consecutive dots (...) & (--), which are often artifacts of text extraction rather than meaningful content.

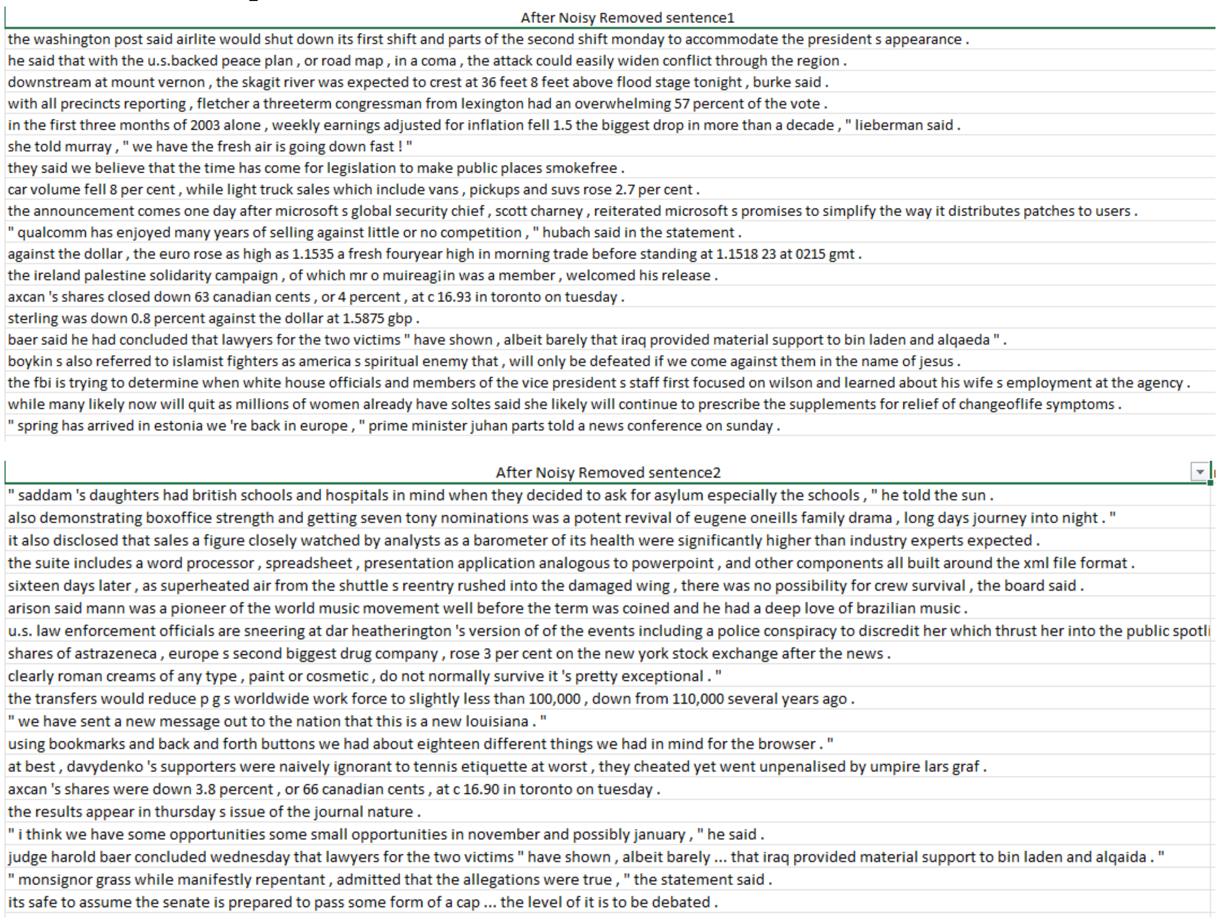
Presence of URLs and Hyperlinks:

- Some sentences include hyperlinks (http://www.smokinggun.com), which do not provide semantic value for paraphrase identification.

Extra Whitespaces:

- Some text samples have leading/trailing spaces due to data collection methods.

4.2.3.2 Snapshot of Dataset After Noise Removal



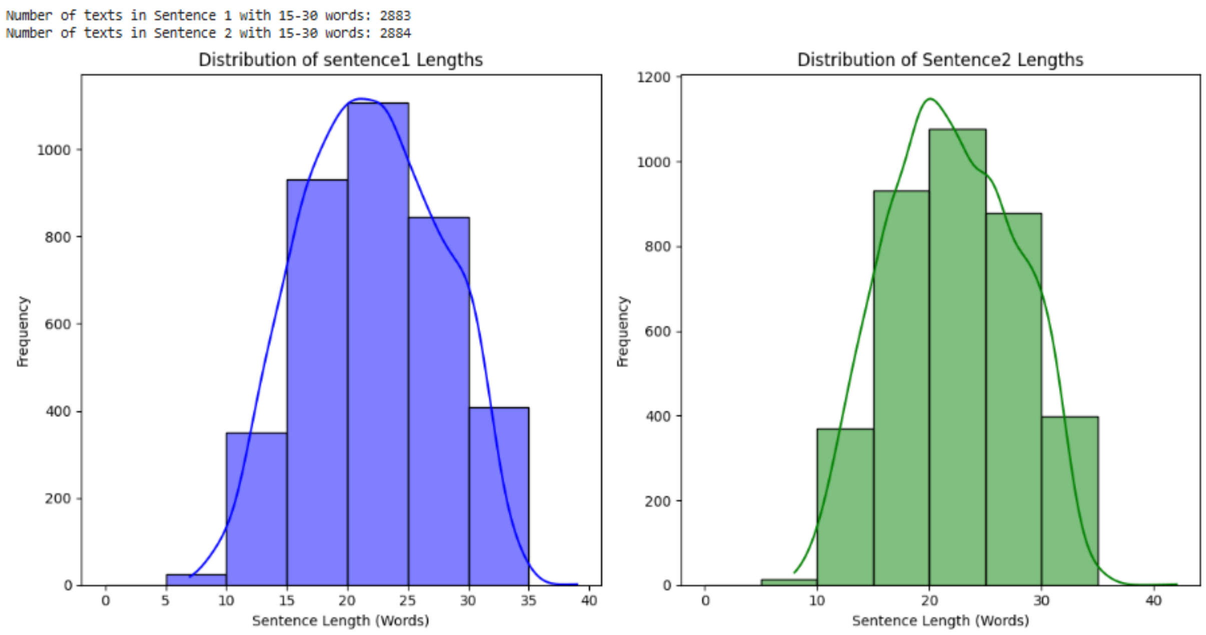
After Noisy Removed sentence1
the washington post said airlite would shut down its first shift and parts of the second shift monday to accommodate the president s appearance .
he said that with the u.s.backed peace plan , or road map , in a coma , the attack could easily widen conflict through the region .
downstream at mount vernon , the skagit river was expected to crest at 36 feet 8 feet above flood stage tonight , burke said .
with all precincts reporting , fletcher a threeterm congressman from lexington had an overwhelming 57 percent of the vote .
in the first three months of 2003 alone , weekly earnings adjusted for inflation fell 1.5 the biggest drop in more than a decade , " lieberman said .
she told murray , " we have the fresh air is going down fast ! "
they said we believe that the time has come for legislation to make public places smokefree .
car volume fell 8 per cent , while light truck sales which include vans , pickups and suvs rose 2.7 per cent .
the announcement comes one day after microsoft s global security chief , scott charney , reiterated microsoft s promises to simplify the way it distributes patches to users .
" qualcomm has enjoyed many years of selling against little or no competition , " hubach said in the statement .
against the dollar , the euro rose as high as 1.1535 a fresh fouryear high in morning trade before standing at 1.1518 23 at 0215 gmt .
the ireland palestine solidarity campaign , of which mr o muireag¡in was a member , welcomed his release .
axcan 's shares closed down 63 canadian cents , or 4 percent , at c 16.93 in toronto on tuesday .
sterling was down 0.8 percent against the dollar at 1.5875 gbp .
baer said he had concluded that lawyers for the two victims " have shown , albeit barely that iraq provided material support to bin laden and alqaeda " .
boykin s also referred to islamist fighters as america s spiritual enemy that , will only be defeated if we come against them in the name of jesus .
the fbi is trying to determine when white house officials and members of the vice president s staff first focused on wilson and learned about his wife s employment at the agency .
while many likely now will quit as millions of women already have soltes said she likely will continue to prescribe the supplements for relief of changeoflife symptoms .
" spring has arrived in estonia we 're back in europe , " prime minister juhan parts told a news conference on sunday .

After Noisy Removed sentence2
" saddam 's daughters had british schools and hospitals in mind when they decided to ask for asylum especially the schools , " he told the sun .
also demonstrating boxoffice strength and getting seven tony nominations was a potent revival of eugene oneills family drama , long days journey into night . "
it also disclosed that sales a figure closely watched by analysts as a barometer of its health were significantly higher than industry experts expected .
the suite includes a word processor , spreadsheet , presentation application analogous to powerpoint , and other components all built around the xml file format .
sixteen days later , as superheated air from the shuttle s reentry rushed into the damaged wing , there was no possibility for crew survival , the board said .
arison said mann was a pioneer of the world music movement well before the term was coined and he had a deep love of brazilian music .
u.s. law enforcement officials are sneering at dar heatherington 's version of of the events including a police conspiracy to discredit her which thrust her into the public spotli
shares of astrazeneca , europe s second biggest drug company , rose 3 per cent on the new york stock exchange after the news .
clearly roman creams of any type , paint or cosmetic , do not normally survive it 's pretty exceptional . "
the transfers would reduce p g s worldwide work force to slightly less than 100,000 , down from 110,000 several years ago .
" we have sent a new message out to the nation that this is a new louisiana . "
using bookmarks and back and forth buttons we had about eighteen different things we had in mind for the browser . "
at best , davydenko 's supporters were naively ignorant to tennis etiquette at worst , they cheated yet went unpenalised by umpire lars graf .
axcan 's shares were down 3.8 percent , or 66 canadian cents , at c 16.90 in toronto on tuesday .
the results appear in thursday s issue of the journal nature .
" i think we have some opportunities some small opportunities in november and possibly january , " he said .
judge harold baer concluded wednesday that lawyers for the two victims " have shown , albeit barely ... that iraq provided material support to bin laden and alqaida . "
" monsignor grass while manifestly repentant , admitted that the allegations were true , " the statement said .
its safe to assume the senate is prepared to pass some form of a cap ... the level of it is to be debated .

Fig. 4.5: Snapshot of Dataset After Noise Removal

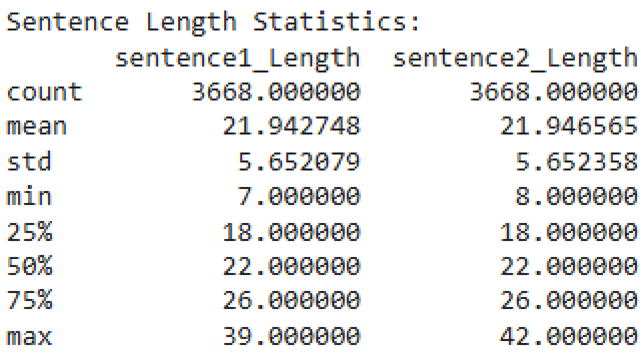
4.2.4 Univariate Analysis

I used a distribution plot to check the data distribution, a box plot to identify outliers, and printed the data description to verify the range of sentence lengths.



```
Number of texts in Sentence 1 with 15-30 words: 2883
Number of texts in Sentence 2 with 15-30 words: 2884
```

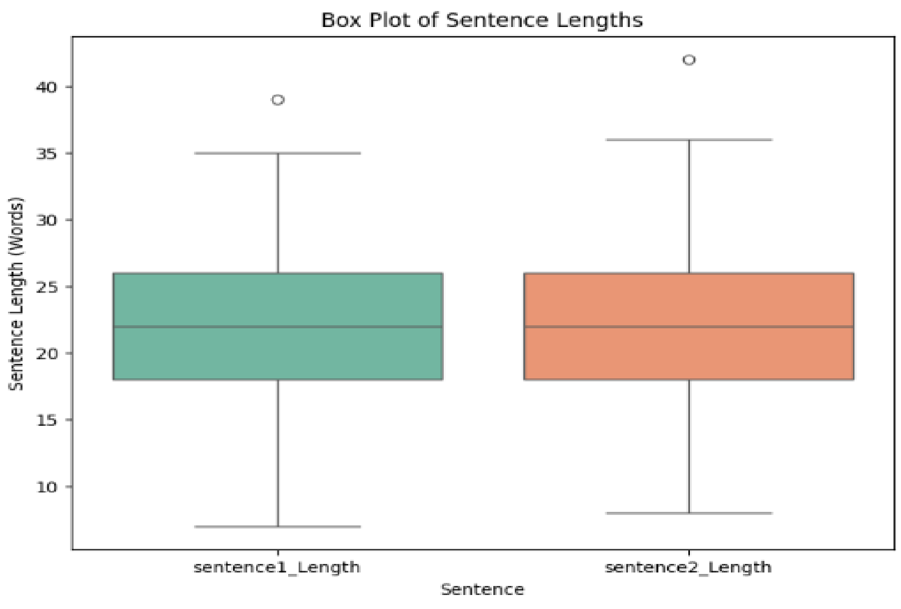
Fig. 4.6: Distribution Plots of Sentence Lengths



```
Sentence Length Statistics:
       sentence1_Length  sentence2_Length
count       3668.000000       3668.000000
mean          21.942748         21.946565
std            5.652079          5.652358
min            7.000000          8.000000
25%           18.000000         18.000000
50%           22.000000         22.000000
75%           26.000000         26.000000
max           39.000000         42.000000
```

Fig. 4.7: Train Set Sentence Length Statistics

The analysis of sentence length distribution provides key insights into the dataset's structure. Based on the statistical summary and visualizations, the sentence lengths for both sentence1 and sentence2 follow an approximately normal distribution. The histogram plots indicate that most sentences contain between 15 to 30 words, which accounts for 0.78% (2883) of the total data. This observation is further confirmed by the statistical summary, where the sentence lengths range from 7 to 39 words for sentence1 and from 8 to 42 words for sentence2. The mean sentence length for both is around 21.94 words, with a standard deviation of approximately 5.65 words.

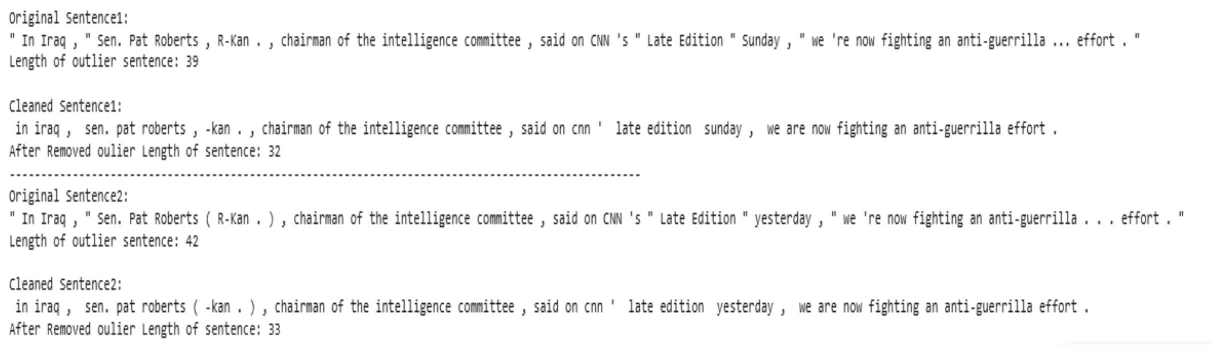


```
Number of Outliers in Sentence Lengths:
sentence1_Length    1
sentence2_Length    1
dtype: int64
```

Fig. 4.8: Box plot for outlier detection

The box plot analysis reveals that the distribution is fairly symmetrical, with a median sentence length of 22 words. Outliers are minimal, with only a single instance exceeding the upper whisker threshold. This suggests that the dataset maintains consistency in sentence length, which is crucial for paraphrase identification models. The limited number of extreme values ensures that the model is less likely to be influenced by exceptionally long or short sentences.

Handling Outliers



```
Original Sentence1:
" In Iraq , " Sen. Pat Roberts , R-Kan . , chairman of the intelligence committee , said on CNN 's " Late Edition " Sunday , " we 're now fighting an anti-guerrilla ... effort . "
Length of outlier sentence: 39

Cleaned Sentence1:
 in iraq ,  sen. pat roberts , -kan . , chairman of the intelligence committee , said on cnn '  late edition  sunday ,  we are now fighting an anti-guerrilla effort .
After Removed oulier Length of sentence: 32
---------------------------------------------------------------------------------------------
Original Sentence2:
" In Iraq , " Sen. Pat Roberts ( R-Kan . ) , chairman of the intelligence committee , said on CNN 's " Late Edition " yesterday , " we 're now fighting an anti-guerrilla . . . effort . "
Length of outlier sentence: 42

Cleaned Sentence2:
 in iraq ,  sen. pat roberts ( -kan . ) , chairman of the intelligence committee , said on cnn '  late edition  yesterday ,  we are now fighting an anti-guerrilla effort .
After Removed oulier Length of sentence: 33
```

Fig. 4.9: Snapshot of Before and After Outlier Handle

A box plot was used to identify outliers, which revealed that only one row in the dataset has outliers. Upon closer inspection, this outlier was found to contain special characters (e.g., multiple consecutive dots (...), multiple double quotes (“)). After removing noise, the sentence lengths were reduced, and the outlier was no longer present.

4.3 Prompt Analysis

In the process of implementing prompt-based learning for paraphrase identification, several key observations and challenges were identified, particularly when working with different models such as OpenAI's GPT-3.5-Turbo, GPT-4o-min, Meta Llama-3.2-1B, Llama-3.2-3B, Mistral-7B and T5. Below is a detailed analysis of the findings and practical challenges encountered during prompt engineering and hyperparameter tuning.

Limitations of Encoder-Based Models for Prompt-Based Learning

One of the key challenges observed was that prompt-based learning is inherently designed for decoder-only or encoder-decoder models, as they are optimized for generative tasks. Encoder-only models (e.g., ModernBERT) lack the ability to effectively interpret and generate responses from prompts because they are primarily designed for feature extraction and classification tasks rather than text generation.

In contrast, decoder-only (e.g., GPT-3.5-Turbo, Llama3.2-1B) and encoder-decoder (e.g., T5) models are more naturally suited for handling prompt-based learning, as they can interpret natural language instructions and generate meaningful responses based on the given input.

4.3.1 Prompts for Different Models

It was observed that OpenAI models (e.g., GPT-3.5-Turbo, GPT-4o-mini), Meta models (e.g., Llama-3.2-1B, Lama-3.2-3B), Mistral-7B and T5 require different prompt structures to achieve optimal performance.

OpenAI GPT Models Prompt:

```
prompt = f"""

We have two sentences. Determine if they are paraphrases of each other.

A paraphrase conveys the same meaning using different words while maintaining the core information. A non-paraphrase has a different meaning or significantly alters the information.

Respond strictly with one word: **paraphrase** or **non-paraphrase**.

    Now classify:

    Sentence 1: "{sentence1}"

    Sentence 2: "{sentence2}"

    Answer:    """
```

Llama-3.2-1B, Llama-3.2-3B and Mistral-7B Prompt : Require more explicit instructions or slightly different phrasing to generate accurate responses.

```
template="""

Determine if two sentences convey the same core information. A paraphrase maintains the essential meaning despite different wording.

Respond with one word: paraphrase or non-paraphrase.

    Now classify:

    Sentence 1: "{sentence1}"

    Sentence 2: "{sentence2}"

    Answer:  """
```

T5 Prompt:

```
template= """Is the following pair of sentences a paraphrase? Sentence 1: {sentence1} Sentence 2: {sentence2} Answer:"""
```

All models are generating a lot of irrelevant text, highlighting the crucial role of prompt engineering in this approach. In the case of encoder-decoder models like T5, the prompt is extremely minimal, directly asking a yes/no question. However, decoder-based models require a more detailed explanation in the prompt to ensure clarity. The phrase "Respond strictly with one word: **paraphrase** or **non-paraphrase**." was included in the GPT-based prompt to minimize unwanted text, as the model tended to generate excessive information. Similarly, in Llama and Mistral, the instruction "Respond with one word: paraphrase or non-paraphrase." was used to enforce concise outputs and reduce irrelevant responses. Few-shot prompting (provided 1–2 examples) was applied, but the response remained almost the same, showing no significant improvement in classification accuracy.

4.3.2 Hyperparameter tuning

4.3.2.1 Temperature (Temp) Tuning

Temperature (Temp) was a critical hyperparameter in prompt-based learning, controlling the randomness of the model's output.

Higher Temperature Values (e.g., 0.7–1.0):

- Generate more creative and diverse responses.
- However, they also increase the likelihood of producing irrelevant or random text.

Lower Temperature Values (e.g., 0.1–0.5):

- Produced more deterministic and focused responses.
- This was particularly useful for tasks like paraphrase identification, where accuracy is more important than creativity.

4.3.2.2 Max New Tokens Tuning

The max_new_tokens parameter controls the length of the generated text.

Higher Values:

- Allowed the model to generate longer responses.
- However, they also lead to garbage text or irrelevant information, especially when model started to diverge from the task.

Lower Values:

- Restricted the response length, ensuring concise and task-relevant outputs.

- However, they sometimes truncated useful information, which affected the completeness of the response.

4.3.3 Model Availability, Performance, and Cost Analysis

Table 4.1: Model Availability, Performance, and Cost Analysis

Model	GPT-3.5-Turbo	GPT-4o-Mini	Llama-3.2-1B	Llama-3.2-3B	Mistral-7B	T5
Access Method	OpenAI API	OpenAI API	HF Download	HF Download	HF Download	HF Download
Optimal Temperature	0.5	0.5	0.2	0.3	0.1	0.1
Optimal Max New Tokens	4	4	4	4	6	5
Avg. Execution Time (mins)	~21.27	~26.38	~3	~4.3	~29	~14
Cost Type	API Cost	API Cost	Cloud Cost	Cloud Cost	Cloud Cost	Cloud Cost
Cost per Execution ($)	~0.14	~0.04	~0.02	~0.032	0.21	~0.1
Total Cost to Tune ($)	~0.90	~0.40	~0.21	~0.33	~2.10	~0.50
Memory Usage (GB)	N/A	N/A	~5 GB	~8 GB	~15 GB	~5 GB
Fine-Tuning Support	No	No	Yes	Yes	Yes	Yes

After conducting multiple rounds of execution, the optimal temperature for each model was determined based on performance metrics such as accuracy and stability of responses. The total cost to tune a model represents the cumulative expense incurred during this process calculated by executing the model multiple times (e.g., 10 iterations) to identify the best-performing temperature setting. Additionally, execution time varied significantly across models, with API-based models (GPT-3.5-Turbo, GPT-4o-mini) taking longer due to processing latency, while locally hosted models (Llama-3.2-1B, Llama-3.2-3B, Mistral-7B, T5) demonstrated faster inference but required sufficient computational resources. Cost per execution was notably higher for API-based models due to usage-based pricing, whereas self-hosted models incurred cloud computing costs rather than direct API charges. Overall, hyperparameter tuning, including temperature and max new tokens, played a crucial role in optimizing performance while balancing cost, execution time, and computational efficiency.

4.4.4 Challenges Faced During Prompt Engineering

- The performance of prompt-based models was highly sensitive to the wording and structure of the prompt.
- Small changes in the prompt led to significant variations in the model's output, which required extensive experimentation to identify the most effective prompt structure.
- Different models like GPT, Llama, Mistral, T5 acted differently even when given the same prompt. This meant we had to make specific changes for each model, making the experimentation process more complicated.
- Finding the optimal temperature (temp) value required extensive experimentation. During the research, it was observed that higher temperature values often led to inconsistent or irrelevant outputs, making it difficult to rely on the model's responses. On the other hand, lower temperature values provided more stable and accurate results, but sometimes made the model overly rigid, especially in cases where slight variations in responses were acceptable.
- Setting an appropriate “max_new_tokens” value was a significant challenge. While lower values ensured concise outputs, they occasionally cut off important parts of the response, leading to incomplete or misleading results. On the other hand, higher values often resulted in irrelevant or redundant text, making it difficult to extract meaningful information.

4.4 Fine-Tuning Analysis

Fine-tuning LLMs for paraphrase identification was challenging due to computational limits, adjusting parameters, and optimizing the models. Managing resources was tough because models like Llama 3.2-3B and Mistral-7B needed a lot of memory. Even with AWS GPU instances, scaling up fine-tuning for bigger models was a significant challenge.

4.4.1 Hyperparameter Tuning

During the fine-tuning process, several key hyperparameters were optimized to achieve the best performance. Below are the details of the hyperparameters and the challenges faced during their tuning:

4.4.1.1 Learning Rate

- The learning rate controlled how quickly the model adapted to the task-specific data.
- It was observed that a too high learning rate caused the model to diverge, while a too low learning rate resulted in slow convergence.
- After experimentation, a learning rate in the range of 1e-5 to 3e-5 was found to be optimal for fine-tuning LLMs.

4.4.1.2 Batch Size

- The batch size determined the number of samples processed before updating the model's weights.
- Larger batch sizes improved training stability but required more memory.
- A batch size of 4 to 32 was used, as it provided a good balance between stability and memory usage.

4.4.1.3 Number of Epochs

- The number of epochs controlled how many times the model saw the entire training dataset.
- It was found that too many epochs led to overfitting, while too few epochs resulted in underfitting.
- Early stopping was implemented to determine the optimal number of epochs, which typically ranged between 4 to 10 epochs for the MRPC dataset.

4.4.1.4 Weight Decay

- Weight decay was used as a regularization technique to prevent the model's weights from growing too large.
- A weight decay value in the range of 0.01 to 0.1 was found to be effective in preventing overfitting.

4.4.1.5 Dropout

- Dropout is a regularization technique used to prevent overfitting by randomly deactivating a fraction of neurons during training.

- Higher dropout rates improve generalization but may slow convergence by reducing the model's learning capacity.
- A dropout rate between 0.05 and 0.12 was used to maintain a balance between preventing overfitting and ensuring stable training.

4.4.1.6 Low-Rank Size

- Low-rank adaptation reduces the number of trainable parameters by approximating weight matrices with lower-rank representations.
- A smaller low-rank size limits the model's capacity to adapt, while a larger size increases computational cost.
- A low-rank size between 8 and 32 was used to balance model adaptability and computational efficiency.

4.4.2 Optimized Training Configurations, Cost, and Accessibility

Table 4.2: Optimized Training Configurations, Cost, and Accessibility

Model Name	Llama-3.2-1B	Llama-3.2-3B	Mistral-7B	Modern BERT	T5-Base
Access Method	HF Download	HF Download	HF Download	HF Download	HF Download
Cost Type	Cloud Cost	Cloud Cost	Cloud Cost	Cloud Cost	Cloud Cost
Single GPU Used	NVIDIA A5000	NVIDIA A5000	NVIDIA A100	NVIDIA RTX5000	NVIDIA RTX5000
Memory Usage (GB)	~6 GB	~12 GB	~32 GB	~3 GB	~6 GB
Prompt-Based Support	Yes	Yes	Yes	No	Yes
Optimal Learning Rate	2.00E-04	2.00E-05	2.00E-05	2.00E-05	1.00E-05
Optimal Batch Size	16	16	32	8	8
Optimal LoRA Dropout	0.07	0.1	0.1	-	-
Optimal Low Rank Size	16	16	8	-	-
Optimal LoRA Scaling	32	32	32	-	-

Execution Time (mins)	~22	~60	~90	~11	~16
Cost per Execution ($)	~0.30	~0.50	~1.25	~0.083	~0.12
Total Cost (Tuning) ($)	~15	~20	~5	~1	~1.4

The models were fine-tuned using cloud-based infrastructure, with access via HF downloads and execution costs dependent on cloud GPU resources. Memory usage varied significantly, with smaller models like ModernBERT (~3GB) requiring less GPU memory, while Llama-3.2-3B (~12GB) and Mistral-7B (~32GB) demanded high-memory GPUs for efficient execution. Execution times increased proportionally with model size, with ModernBERT (~11 mins) requiring the least time, whereas Mistral-7B (~90 mins) and Llama-3.2-3B (~60 mins) had significantly longer runtimes due to their larger parameter sizes and memory requirements. The total cost of fine-tuning varied based on model complexity and execution time, with ModernBERT being the most cost-effective, while Llama-3.2-3B and Mistral-7B incurred higher expenses. The high execution cost per run of Mistral-7B made repeated tuning expensive, necessitating careful hyperparameter selection to minimize costs.

4.4.3 Challenges in Model Fine-Tuning and Optimization

4.4.3.1 Hardware Limitations and Resource Constraints

One of the biggest challenges faced during fine-tuning was the high GPU memory requirement of large-scale transformer models. Training on large datasets can be time-consuming and expensive, especially for researchers with limited access to hardware.

- Using AWS g4dn.xlarge (single GPU Nvidia T4, 16 GB GPU memory), I was unable to fine-tune Llama 3.2 (3B) or Mistral-7B due to out-of-memory (OOM) errors.
- Upgrading to AWS ml.g5n.xlarge (single GPU Nvidia A10G, 24 GB GPU memory) allowed fine-tuning of Llama-3.2-1B but was still insufficient for larger models like Mistral-7B.
- Even with gradient checkpointing and mixed-precision training (FP16/BF16), larger models exceeded available GPU memory, making it impractical to fine-tune them without distributed training across multiple GPUs.

4.4.3.2 Limitations of LoRA and QLoRA Approaches

Since full fine-tuning was not feasible, LoRA and QLoRA were used to efficiently update only specific parts of the model while keeping most parameters frozen. However, these approaches had their own limitations:

Challenges in LoRA Fine-Tuning:

- Underfitting: If LoRA rank (r) was too low, the model failed to adapt well to paraphrase detection.
- Trade-off Between Efficiency and Performance: Higher r values improved learning but increased memory consumption, making it challenging to find the optimal setting.

Challenges in QLoRA Implementation:

- Quantization Loss: Reducing model precision to 4-bit caused slight performance degradation, especially in handling subtle paraphrase variations.
- Longer Training Time: While it saves memory, QLoRA requires longer training time to reach optimal performance.

4.4.3.3 Difficulties in Optimizing Hyperparameters

- Finding the optimal combination of hyperparameters required extensive experimentation, which was time-consuming and computationally expensive.
- Multiple iterations were performed to test different values of learning rate, batch size, and number of epochs.
- The MRPC dataset is relatively small (5801 pairs), which increased the risk of overfitting.
- Techniques like early stopping, dropout, and regularization were implemented to mitigate this issue.
- Fine-tuning large models like Llama-3.2-3B, Mistral-7B required substantial GPU memory, which was a bottleneck.
- Techniques like gradient checkpointing and mixed precision training were used to reduce memory usage.

- Despite these optimizations, fine-tuning Mistral-7B was not feasible on single-GPU instances and required multi-GPU training and high-memory.

4.4.3.4 Strategies to Overcome Challenges

Resource Optimization and Scaling Strategies:

- Fine-Tuning LLMs on Multi-GPU Systems: JarvisLabs AI provides access to high-performance GPUs such as the Nvidia A100 and A5000, which offer significantly more memory and computational power compared to the T4 and A10G GPUs.
- LoRA: Helped reduce trainable parameters but still required significant memory for models above 3B parameters.
- QLoRA: Used 4-bit quantization to reduce GPU memory usage, but led to slight precision loss, impacting final performance.
- Gradient Accumulation: Allowed smaller batch sizes but significantly increased training time.
- Using CPU Offloading: Tried model offloading techniques to shift computations between GPU and CPU, but it slowed down training considerably.

Hyperparameter Optimization Strategies

- Extensive experimentation was conducted to find the optimal combination of hyperparameters (e.g., learning rate, batch size, number of epochs).
- Early stopping was implemented to prevent overfitting and determine the optimal number of epochs.
- Techniques like weight decay and dropout were used to prevent overfitting and improve generalization.
- Techniques like gradient checkpointing and mixed precision training were used to reduce memory usage during fine-tuning.

4.5 Summary

The Analysis & Design phase provided critical insights into dataset structure, model performance, and optimization strategies. The MRPC dataset was carefully examined, and appropriate preprocessing techniques were applied to enhance data quality and ensure optimal input for model training. The study demonstrated that prompt-based learning is not well-suited for encoder-based models like ModernBERT, as they require fine-tuning instead of prompt engineering for effective results. Conversely, decoder-based models such as GPT-3.5-Turbo, Llama-3.2-1B, Llama-3.2-3B, Mistral-7B and T5 benefit significantly from well-designed prompts and hyperparameter adjustments, such as temperature tuning and max new tokens settings. In terms of fine-tuning, LoRA and QLoRA were explored to reduce memory requirements while maintaining performance. However, resource constraints posed significant challenges, particularly for large-scale models like Mistral-7B, which could not be fine-tuned effectively on single-GPU instances due to high memory demands. Techniques like gradient accumulation, mixed-precision training, and CPU offloading were implemented to manage memory usage, but they also introduced trade-offs in training efficiency.

Additionally, this chapter analyzed model availability, computational efficiency, and cost, highlighting the trade-offs between API-based and self-hosted models. The findings emphasize the importance of selecting an optimal balance between performance, resource utilization, and cost-effectiveness when working with LLMs.The next chapter, Results & Discussion, will focus on evaluating the fine-tuned models and base transformer models using key evaluation metrics such as accuracy, F1-score, etc.. A comparative analysis will be conducted to determine the effectiveness of different models and optimization strategies in achieving high-quality paraphrase identification.

CHAPTER 5

RESULTS AND DISCUSSIONS

5.1 Introduction

This chapter presents the results and discussions of experiments conducted for paraphrase identification using two approaches: prompt-based learning and fine-tuned (decoder-based modeles, encoder-based models and encoder-decoder based models). The experiments evaluated various models, including OpenAI's GPT-3.5 Turbo, GPT-4o Mini, Meta's Llama-3.2-1B-Instruct, Llama-3.2-3B-Instruct, Mistral-7B-Instruct, Google's T5-base and ModernBERT. The findings compare the performance of these models, focusing on accuracy, computational efficiency, and cost. The results highlight the strengths and limitations of each approach, providing a comprehensive understanding of their applicability to paraphrase identification tasks.

5.2 Prompt-Based Models Performance Evaluation and Results

Initially, paraphrase identification was conducted using a prompt-based approach with various LLM models, including OpenAI's GPT-3.5-Turbo, GPT-4o-mini, Meta's Llama-3.2-1B, Llama-3.2-3B, Mistral-7B and T5. The experiments were conducted with different temperature settings and sample sizes to best model performance in varying conditions. Some experiments were conducted on a smaller sample size to optimize cost and processing time. If the results from the small sample size were satisfactory, the evaluation was then extended to the entire dataset. The key evaluation metrics accuracy, precision, recall, and F1 score were analyzed to compare the performance of different models.

Performance Evaluation

To determine the best-performing models, multiple experiments were conducted by varying the temperature settings and sample sizes. After testing different configurations, the optimal results for each model were identified. The following table summarizes the best-performing results for each prompt-based model.

Table 5.1: Performance comparison between prompt-based models

Evaluation Metrics	gpt-3.5-turbo	gpt-4o-mini	Llama-3.2-1B-Instruct	Llama-3.2-3B-Instruct	Mistral-7B-Instruct-v0.3	T5-base
Accuracy	0.69	0.74	0.67	0.54	0.76	0.58
Precision	0.88	0.89	0.68	0.83	0.74	0.69
Recall	0.61	0.70	0.96	0.39	0.99	0.65
F1 Score	0.72	0.78	0.79	0.53	0.85	0.67

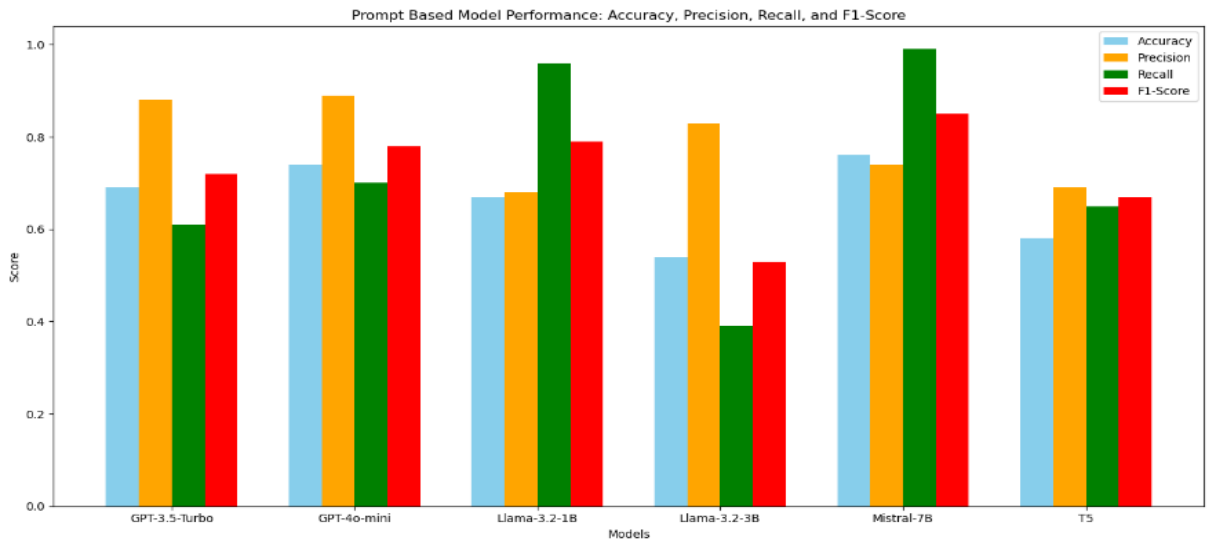


Fig. 5.1: Prompt Based Model Performance: Accuracy, Precision, Recall, and F1-Score

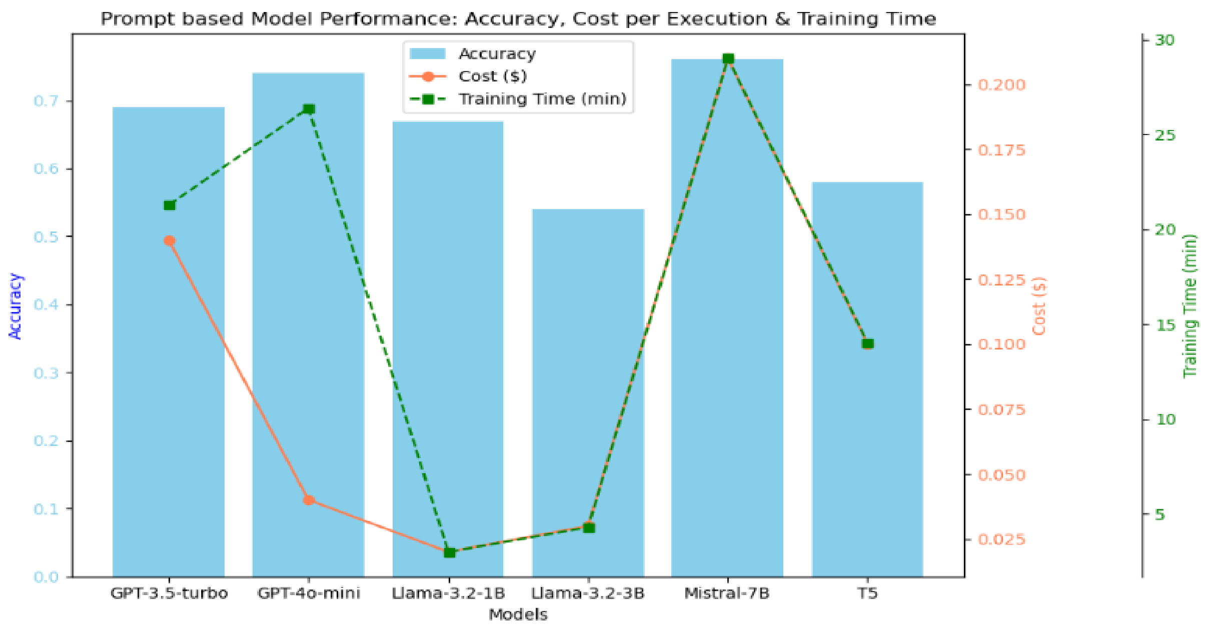


Fig. 5.2: Prompt Based Model Performance: Accuracy, Cost per Execution & Training Time

Choosing the right model involves finding a balance between computational efficiency, cost, and performance. For instance, GPT-4o-Mini reached an accuracy of 0.74 with a cost of $0.04 per execution, making it a cost-effective choice for large-scale tasks. However, its execution time 26.38 minutes to process the full dataset suggests it may not be suitable for real-time applications. On the other hand, open-source models like Llama-3.2-1B-Instruct, Llama-3.2-3B-Instruct, and Mistral-7B-Instruct were run on an NVIDIA A5000 GPU. These models avoided direct API costs but required higher computational resources. Among them, Mistral-7B-Instruct achieved the highest recall (0.99) and F1 score (0.85), but it took 29 minutes to process, making it computationally expensive.

When it comes to temperature tuning costs, the cumulative costs of running multiple experiments including the model download. OpenAI models like GPT-3.5-Turbo and GPT-4o-mini incurred API costs, while Llama-3.2-1B, Llama-3.2-3B, and Mistral-7B, running on GPUs, incurred computational expenses instead. Fine-tuning experiments showed that Llama-3.2-1B-Instruct provided the best trade-off between cost and accuracy, significantly improving performance compared to basic prompt-based use. Larger models, such as Mistral-7B, showed potential but came with significant computational challenges. In conclusion, the choice of model depends on the specific trade-offs between cost, computational efficiency, and accuracy. GPT-4o-Mini stands out for API-based tasks, while Llama-3.2-1B is an excellent choice for fine-tuning in a controlled computational environment.

5.3 Fine-Tuned Model Results

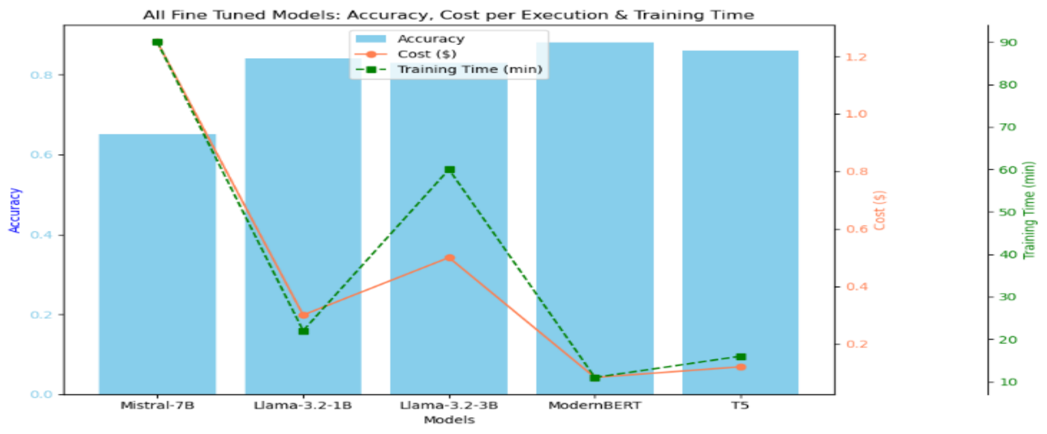


Fig. 5.3: Fine-tuned Based Model Performance: Accuracy, Cost per Execution & Training Time

ModernBERT achieved the highest accuracy (0.88) with the lowest cost ($0.083) and shortest training time (11 mins), making it the most efficient model. Llama-3.2-1B performed well (0.84 accuracy) at a moderate cost ($0.30) and training time (22 mins). T5 provided a balanced trade-off with 0.86 accuracy at $0.12 cost and 16 mins training. Llama-3.2-3B showed similar accuracy (0.83) but required more resources. Mistral-7B performed the worst (0.65 accuracy) with the highest cost ($1.25) and longest training time (90 mins). Below, find the detailed description of each model type and their performance.

5.3.1 Decoder-Based Models Performance Evaluation

Following the prompt-based evaluation, fine-tuning experiments were conducted using Llama-3.2-1B, Llama-3.2-3B, and Mistral-7B-Instruct, models. The experiments involved extensive hyperparameter tuning, including adjustments to learning rate, LoRA dropout, LoRA rank, LoRA alpha, batch size, and other parameters. After testing multiple configurations, the best results were obtained as summarized below.

Table 5.2: Performance comparison between Decoder-based models

Model Name	Llama3.2-1B-Instruct	Llama3.2-3B-Instruct	Mistral-7B-Instruct
Accuracy	0.841	0.83	0.65
Precision	0.892	0.87	0.741
Recall	0.865	0.85	0.745
F1-Score	0.878	0.86	0.743

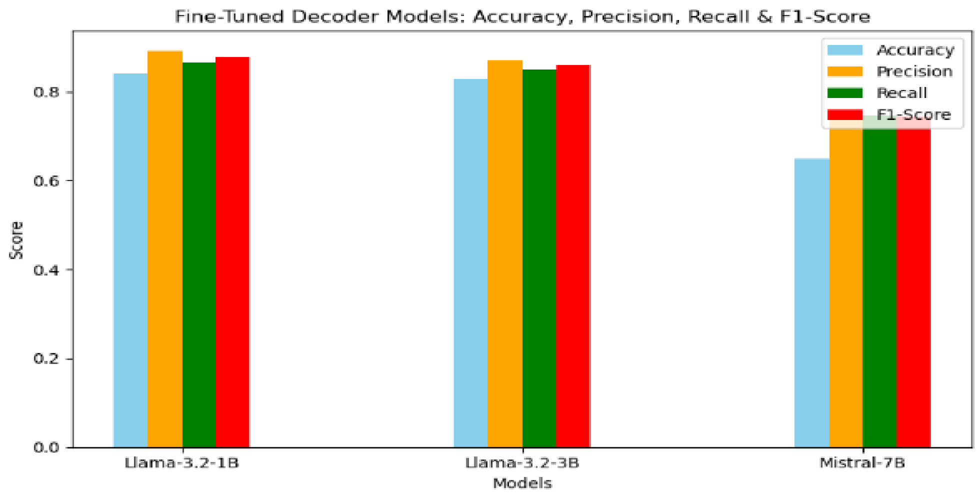


Fig. 5.4: Fine Tuned Decoder-based Model Performance

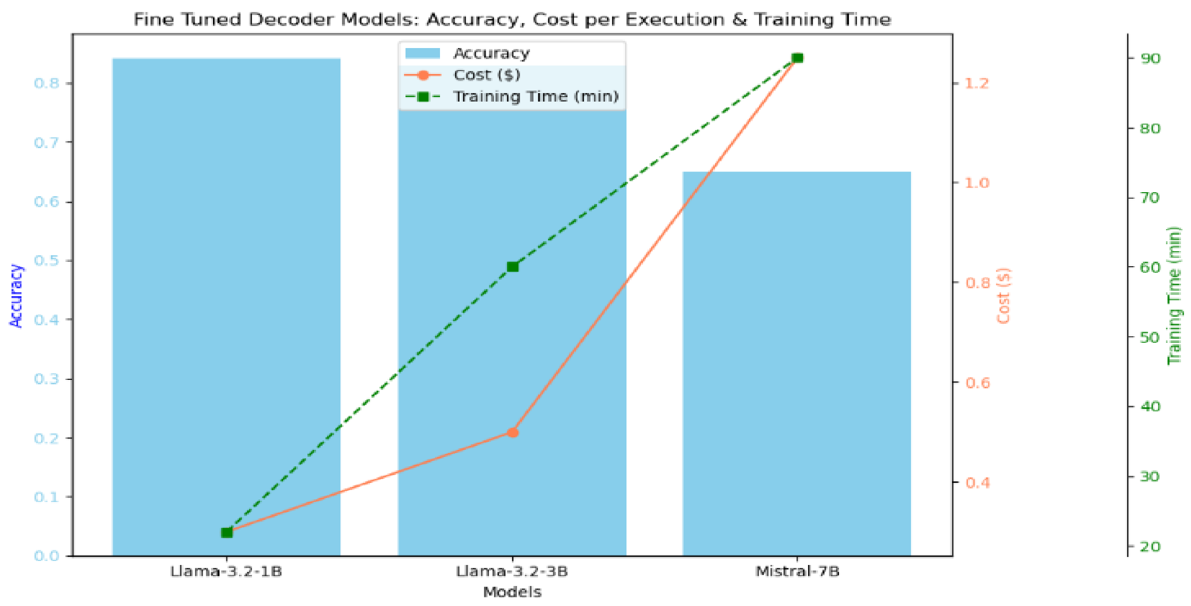


Fig. 5.5: Fine Tuned Decoder-based Models: Accuracy, Cost & Training Time

Fine-tuning significantly enhanced model performance compared to prompt-based inference. Llama-3.2-1B achieved 0.841 accuracy and 0.878 F1-score, while Llama-3.2-3B reached 0.83 accuracy and 0.86 F1-score. The training process required meticulous tuning of LoRA and QLoRA parameters, including dropout, rank size, and scaling factor, to optimize performance. Among the models, Llama-3.2-1B was the most computationally efficient, completing fine-tuning in 22 minutes at a cost of $0.30 per execution, whereas Llama-3.2-3B required 60 minutes with a $0.50 execution cost. The total fine-tuning expenditure was approximately $15 and $20, respectively, accounting for multiple iterations to identify the best hyperparameters. This iterative tuning process involved experimenting with different learning rates, batch sizes, and dropout values to achieve optimal results.

In contrast, Mistral-7B-Instruct posed significant computational challenges. While achieving a 0.65 accuracy and 0.743 F1-score, its fine-tuning process proved prohibitively expensive. On a single NVIDIA A100 GPU (7 CPUs | 32GB RAM | 40GB VRAM), fine-tuning took 1 hour 5 minutes. On a single A5000 GPU (32 CPUs | 64GB RAM | 24GB VRAM), the execution time increased to 2 hours 15 minutes. With a $1.25 execution cost, the projected total fine-tuning cost was $5 before the process was discontinued due to excessive computational demands. This highlights the challenge of fine-tuning larger models, which often require extensive GPU resources, making them impractical for iterative optimization unless substantial computational power is available.

Overall, fine-tuning significantly improved model accuracy, with Llama-3.2-1B emerging as the top performer for domain-specific paraphrase identification. Llama-3.2-3B demonstrated competitive results, but its slightly lower accuracy suggests that a smaller, well-optimized model can be more effective. Although Mistral-7B has the potential for better performance, its limited fine-tuning results were largely due to high computational requirements and restricted GPU resources. This evaluation underscores the trade-offs between model size, accuracy, and computational feasibility. Furthermore, fine-tuning demands multiple iterations for hyperparameter optimization, increasing overall resource consumption. Therefore, selecting the right model requires balancing performance, efficiency, and hardware constraints to achieve optimal results.

5.3.2 Encoder-Based Model

Encoder based experiments were conducted using ModernBERT models. The experiments involved hyperparameter tuning, including adjustments to learning rate, batch size, and other parameters. After testing multiple configurations, the best results were obtained as summarized below.

Table 5.3: Performance of Encoder-based model

Model Name	ModernBERT
Accuracy	0.88
Precision	0.90
Recall	0.92
F1-Score	0.91

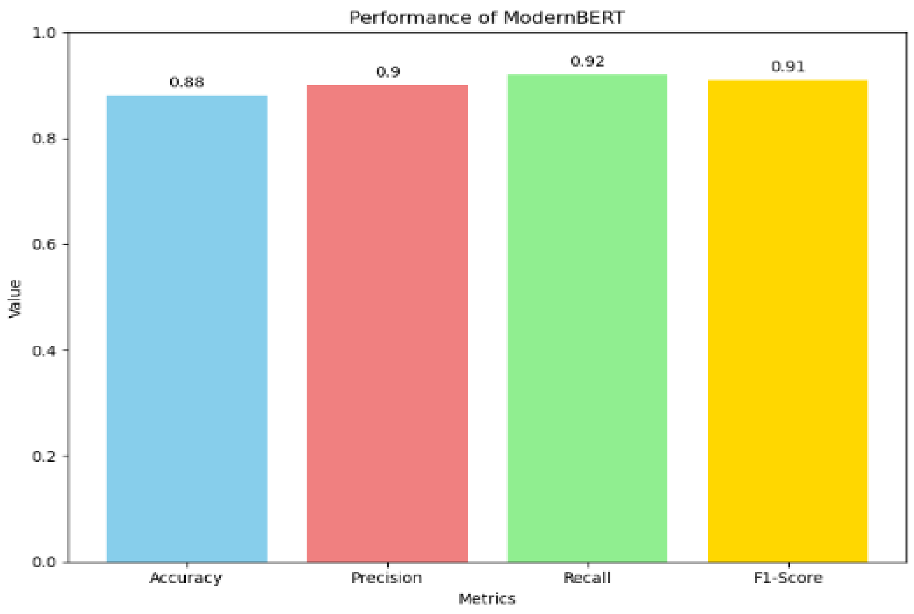


Fig. 5.6: Fine Tuned Encoder-based Model Performance

Fine-tuning ModernBERT demonstrated superior performance in paraphrase identification compared to decoder-based models while being significantly more cost-efficient. ModernBERT achieved an accuracy of 0.88, a precision of 0.90, a recall of 0.92, and an F1-score of 0.91, outperforming Llama-3.2-1B and Llama-3.2-3B in all key metrics. Additionally, it completed fine-tuning in just 11 minutes, with a cost per execution of $0.083, making it the most efficient model in terms of both computational resources and cost-effectiveness. Compared to decoder-based models, which required higher execution times and cost per run, ModernBERT's efficiency highlights the advantages of using encoder-based models for fine-tuning in resource-constrained environments. This evaluation underscores the importance of selecting the right model architecture, as ModernBERT not only reduced computational overhead but also delivered superior results, making it a highly viable option for paraphrase detection tasks.

5.3.3 Results and Analysis of Encoder-Decoder-Based Model Performance

Encoder-decoder-based experiments were conducted using T5. The experiments involved hyperparameter tuning, including adjustments to learning rate, batch size, and other parameters. After testing multiple configurations, the best results were obtained as summarized below.

Table 5.4: Performance of Encoder-decoder-based model

Model Name	Google/T5-base
Accuracy	0.86
Precision	0.86
Recall	0.94
F1-Score	0.90

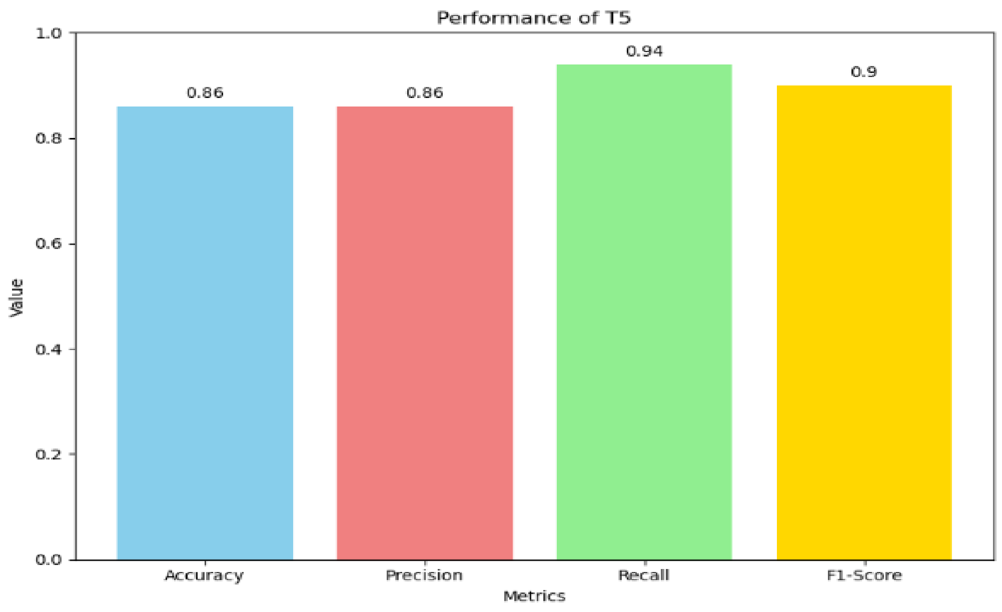


Fig. 5.7: Fine Tuned Encoder-decoder-based Model Performance

Fine-tuning T5 demonstrated strong performance in paraphrase identification while maintaining a balanced trade-off between accuracy and computational cost. T5 achieved an accuracy of 0.86, precision of 0.86, recall of 0.94, and an F1-score of 0.90, making it a competitive choice for paraphrase detection. The model completed fine-tuning in 16 minutes, with a cost per execution of $0.12, which remains lower than decoder-based models like Llama-3.2-1B and Llama-3.2-3B. While its accuracy was slightly lower than ModernBERT, T5 excelled in recall, indicating its strength in correctly identifying paraphrases. This suggests that encoder-decoder architectures like T5 can effectively balance precision and recall while keeping resource utilization manageable. The results further emphasize that decoder-based models are not always the optimal choice for fine-tuning, as models like T5 offer competitive performance at a fraction of the computational cost.

5.4 Comparison between Prompt-based and Fine-tuned based

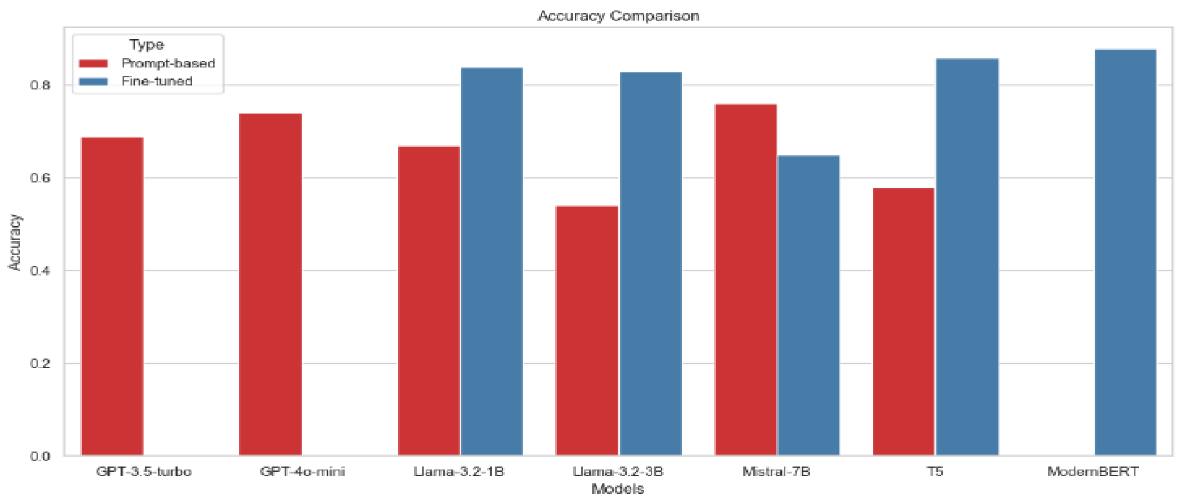


Fig. 5.8: Accuracy Comparison between Prompt-based and Fine-tuned based Models

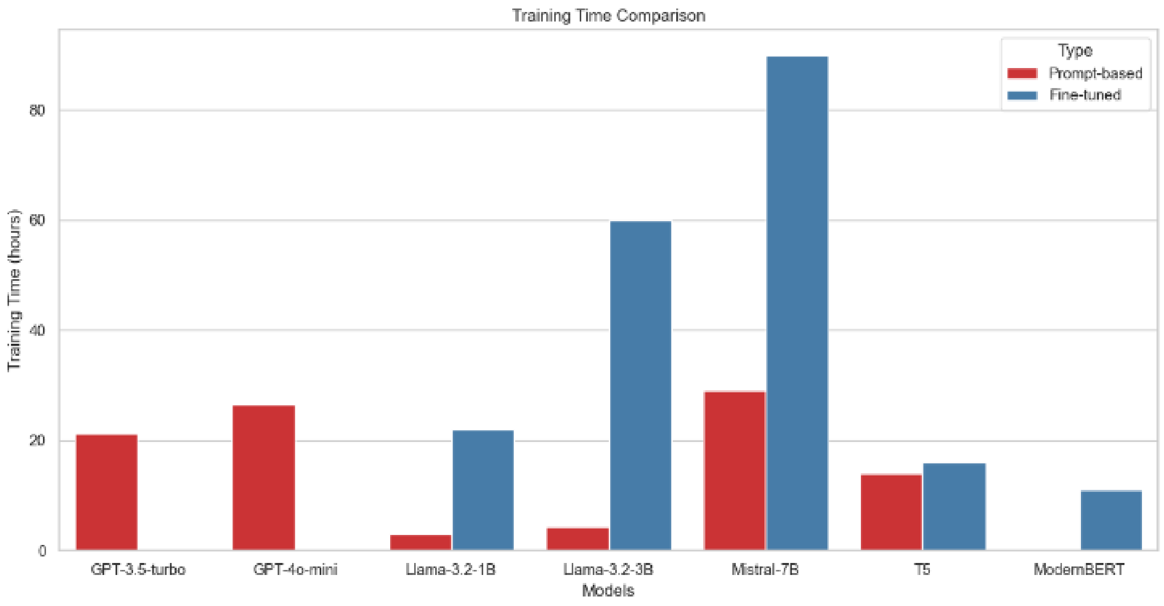


Fig. 5.9: Training Time Comparison between Prompt-based and Fine-tuned based Models

Performance Comparison

Prompt-based models, such as GPT-3.5-Turbo, GPT-4o-Mini, Llama-3.2-1B, and Mistral-7B, were evaluated using API or cloud-based execution. Among them, Mistral-7B achieved the highest accuracy (0.76) and F1-score (0.85), followed by GPT-4o-Mini with 0.74 accuracy and 0.78 F1-score. However, these models demonstrated longer execution times (21–29 minutes per inference) and incurred API-based or cloud computing costs. Additionally, Llama-3.2-3B struggled with accuracy (0.54), indicating potential limitations in prompt-based approaches for domain-specific tasks.

In contrast, fine-tuned models significantly outperformed their prompt-based counterparts. ModernBERT achieved the highest accuracy (0.88) and F1-score (0.91), followed by T5-Base (0.86 accuracy, 0.90 F1-score). Llama-3.2-1B also showed strong performance (0.841 accuracy, 0.878 F1-score), while Mistral-7B underperformed (0.65 accuracy, 0.743 F1-score) due to limited fine-tuning iterations caused by high computational demands. Notably, smaller decoder-based models like Llama-3.2-1B performed significantly better than larger models like Mistral-7B. This suggests that instead of relying on large-scale models, using smaller quantized models offered by various AI providers can be more efficient and effective.

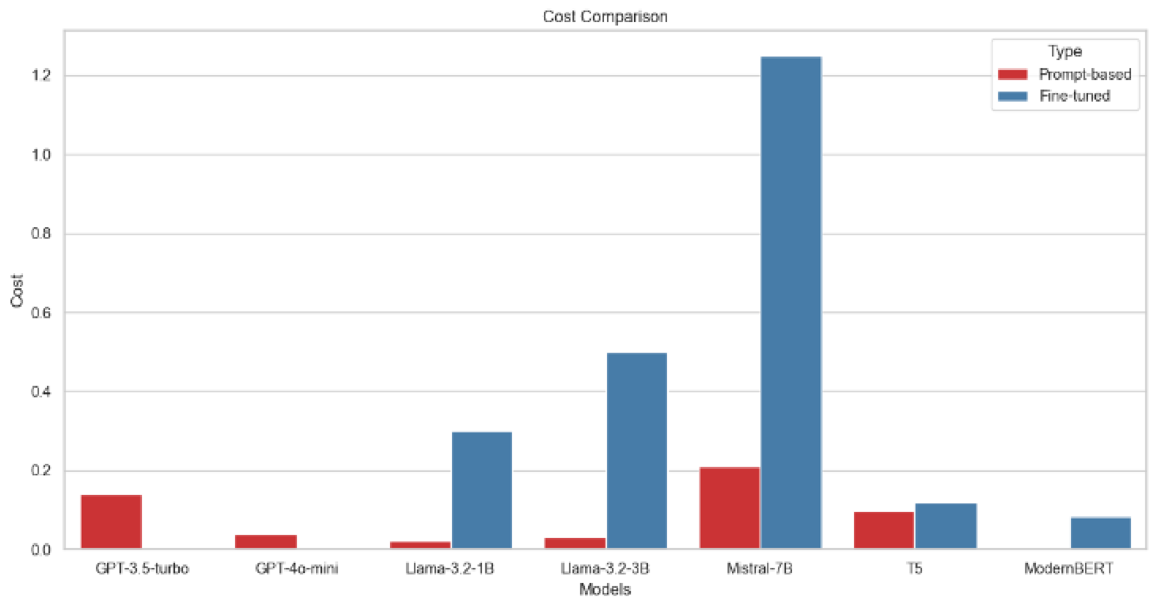


Fig. 5.10: Cost Comparison between Prompt-based and Fine-tuned based Models

Fine-tuning introduced additional expenses due to multiple training iterations, ranging from $0.083 per execution for ModernBERT to $1.25 for Mistral-7B. The total fine-tuning costs varied, with ModernBERT being the most cost-efficient ($1 total cost) and Llama-3.2-3B requiring $20 due to extensive tuning iterations. While prompt-based models required lower per-execution costs ($0.04–$0.21), they lacked adaptability and domain-specific learning capabilities.

5.5 Summary

This chapter provided a comprehensive evaluation of four modeling approaches: prompt-based, fine-tuned decoder models, encoder-based models and encoder-decoder-based, analyzing their accuracy, computational efficiency, and cost. The results indicate that fine-tuned models, such as Llama-3.2-1B, achieved high accuracy but required considerable computational resources. In contrast, encoder-based models like ModernBERT delivered strong performance with lower costs and faster training times. While prompt-based methods were more accessible and cost-effective, they generally underperformed compared to fine-tuned models.

CHAPTER 6

CONCLUSIONS AND RECOMMENDATIONS

6.1 Introduction

This chapter summarizes the key findings of the research, discusses their implications, and provides recommendations for future work. The study explored four approaches for paraphrase identification: prompt-based learning, fine-tuned decoder-based models, encoder-based model and encoder-decoder-based model. Each approach was evaluated based on accuracy, computational efficiency, and cost. The chapter discusses the strengths and limitations of each method, providing insights into their practical applications. This chapter also outlines the contributions to knowledge and suggests future directions to advance the field of paraphrase identification.

6.2 Discussion and Conclusion

6.2.1 Prompt-Based Learning

Strengths: Prompt-based methods are cost-effective and easy to implement, requiring no additional training. Models like GPT-4o Mini achieved competitive accuracy (0.74) with low execution costs ($0.04). These methods provide flexibility for handling diverse datasets without requiring fine-tuning.

Limitations: Prompt-based models are highly sensitive to prompt design and temperature settings, making optimization challenging. Additionally, they generally underperform compared to fine-tuned models and can have higher execution times, limiting their suitability for real-time applications.

6.2.2 Fine-Tuned Based Leaning

6.2.2.1 Decoder-Based Models

Strengths: Fine-tuning significantly enhances model performance, as seen with models like Llama-3.2-1B and Llama-3.2-3B, which achieved high accuracy (0.84) and F1-score (0.88).

These models excel in domain-specific tasks by adapting to specialized datasets, leading to improved contextual understanding.

Limitations: Fine-tuning demands substantial computational resources, increasing both cost and training time. Larger models, such as Mistral-7B, often require extensive hardware, making them impractical for resource-constrained environments. Additionally, fine-tuned models are optimized for a specific dataset and may not generalize well to different tasks.

6.2.2.2 Encoder-Based Models

Strengths: Encoder-based models like ModernBERT offer an optimal balance of accuracy, efficiency, and cost. ModernBERT achieved the highest accuracy (0.88) and F1-score (0.91) with minimal execution time (11 minutes) and low cost ($0.083 per execution). These models are well-suited for structured tasks and require fewer computational resources compared to decoder-based models.

Limitations: While efficient, encoder-based models struggle with imbalanced datasets and may require additional pre-processing for optimal performance. They may also lack the generative capabilities of decoder-based models, limiting their flexibility in certain NLP tasks.

6.2.2.3 Encoder-Decoder-based Models

Strengths: Encoder-decoder models, such as T5, provide a flexible architecture that performs well across diverse NLP tasks. T5 achieved a strong balance between accuracy (0.86) and F1-score (0.90), with a moderate execution time (16 minutes) and cost ($0.12 per execution). These models leverage both encoding and decoding mechanisms, making them versatile for tasks requiring both comprehension and generation.

Limitations: Despite their versatility, encoder-decoder models can be computationally demanding, requiring more resources than encoder-only models. Additionally, they may not match the fine-tuned performance of decoder-based models for domain-specific applications.

6.2.3 Conclusion

This study explored different modeling approaches for paraphrase identification, comparing prompt-based, fine-tuned decoder models, encoder-based models, and encoder-decoder models. The findings highlight that prompt-based methods are cost-effective and flexible, making them suitable for dynamic datasets, but they often struggle to match the accuracy of fine-tuned models. Fine-tuned decoder-based models achieved the highest accuracy but required significant computational resources, making them less practical for environments with hardware constraints. Encoder-based models, such as ModernBERT, provided a strong balance of efficiency and accuracy, making them a viable choice for structured NLP tasks. Encoder-decoder models like T5 demonstrated versatility, performing well across multiple tasks while maintaining reasonable computational efficiency. Ultimately, the choice of model depends on the specific use case. Fine-tuning is ideal for scenarios with a fixed dataset and well-defined tasks, while prompt-based approaches offer flexibility for varying data. For resource-efficient deployment, smaller quantized models are preferable over large-scale models.

6.3 Contribution to knowledge

This research contributes to the growing field of LLM-based paraphrase identification by:

- Comparing decoder-based, encoder-based, and encoder-decoder architectures for paraphrase detection.
- Evaluating fine-tuning strategies, including LoRA & QLoRA, to optimize performance while minimizing GPU memory usage.
- Analyzing computational trade-offs between API-based inference, fine-tuning, and encoder-based models.
- Identifying cost-effective and scalable approaches for real-world paraphrase detection tasks.
- Offering practical insights into selecting models based on accuracy, cost, and computational efficiency.
- Providing a detailed breakdown of execution time and cost, aiding practitioners in making informed deployment decisions.

6.4 Future Recommendations

- Explore hybrid approaches that combine prompt-based and fine-tuned methods, such as using prompt-based models for initial filtering and fine-tuned models for final classification.
- Conduct experiments on larger and more diverse datasets to improve model robustness and generalizability.

- Investigate advanced fine-tuning techniques, such as adapter modules or sparse fine-tuning, to enhance efficiency while maintaining high accuracy.
- Optimize quantization strategies for smaller models to balance performance and resource consumption in real-world applications.

REFERENCES

Al-Jibory, F.K. and Al-Tamimi, M.S.H., (2021) *Hybrid System for Plagiarism Detection on A Scientific Paper*. [online] *Turkish Journal of Computer and Mathematics Education*, Available at: www.gutenberg.org.

Aziz, A.A., Djamal, E.C. and Ilyas, R., (2019) *Paraphrase Detection Using Manhattan's Recurrent Neural Networks and Long Short-Term Memory*.

Aziz, A.A., Djamal, E.C. and Ilyas, R., (2024) *Siamese Similarity Between Two Sentences Using Manhattan's Recurrent Neural Networks*.

Bakar, A., Rahman, S., Thang Ta, H., Najjar, L. and Gelbukh, A., (2022) *Paraphrase Identification: Lightweight Effective Methods Based Features from Pre-trained Models*.

Bojanowski, P., Grave, E., Joulin, A. and Mikolov, T., (2016) Enriching Word Vectors with Subword Information. [online] Available at: http://arxiv.org/abs/1607.04606.

Chavan Hiten, Taufik Mohd., kadave Rutuja and Chandra Nikita, (2021) Plagiarism Detector Using Machine Learning.

Dolan, W.B. and Brockett, C., (2005) *Automatically Constructing a Corpus of Sentential Paraphrases*. [online] Available at: http://www.cogsci.princeton.edu/~wn/.

Dolan, W.B. and Brockett, C., (n.d.) *Automatically Constructing a Corpus of Sentential Paraphrases*.

Eppa, A. and Murali, A.H., (2021) Machine Learning Techniques for Multisource Plagiarism Detection. In: *CSITSS 2021 - 2021 5th International Conference on Computational Systems and Information Technology for Sustainable Solutions, Proceedings*. Institute of Electrical and Electronics Engineers Inc.

Fan, A., Wang, S. and Wang, Y., (2024) Legal Document Similarity Matching Based on Ensemble Learning. *IEEE Access*, 12, pp.33910–33922.

Gahman, N. and Elangovan, V., (2023) A Comparison of Document Similarity Algorithms. *International Journal of Artificial Intelligence & Applications*, 142, pp.41–50.

Gangadharan Veena, A Athira T, L Amritha and Gupta Deepa, (2020) *Paraphrase Detection Using Deep Neural Network Based Word Embedding Techniques*. IEEE.

Gontumukkala, S.S.T., Godavarthi, Y.S.V., Gonugunta, B.R.R.T., Gupta, D. and Palaniswamy, S., (2022) Quora Question Pairs Identification and Insincere Questions Classification. In: *2022*

13th International Conference on Computing Communication and Networking Technologies, ICCCNT 2022. Institute of Electrical and Electronics Engineers Inc.

Hu, E.J., Shen, Y., Wallis, P., Allen-Zhu, Z., Li, Y., Wang, S., Wang, L. and Chen, W., (2021) LoRA: Low-Rank Adaptation of Large Language Models.

Hunt, E., Janamsetty, R., Kinares, C., Koh, C., Sanchez, A., Zhan, F., Ozdemir, M., Waseem, S., Yolcu, O., Dahal, B., Zhan, J., Gewali, L. and Oh, P., (2019) Machine learning models for paraphrase identification and its applications on plagiarism detection. In: *Proceedings - 10th IEEE International Conference on Big Knowledge, ICBK 2019*. Institute of Electrical and Electronics Engineers Inc., pp.97–104.

Islam, A., Rahman, E., Chowdhury, A.A. and Mojumder, M.A.N., (2023) A Deep Learning Approach to Detect Plagiarism in Bengali Textual Content using Similarity Algorithms. In: *Proceedings of IEEE InC4 2023 - 2023 IEEE International Conference on Contemporary Computing and Communications*. Institute of Electrical and Electronics Engineers Inc.

Jiang, A.Q., Sablayrolles, A., Mensch, A., Bamford, C., Chaplot, D.S., Casas, D. de las, Bressand, F., Lengyel, G., Lample, G., Saulnier, L., Lavaud, L.R., Lachaux, M.-A., Stock, P., Scao, T. Le, Lavril, T., Wang, T., Lacroix, T. and Sayed, W. El, (2023a) Mistral 7B. [online] Available at: http://arxiv.org/abs/2310.06825.

Jiang, A.Q., Sablayrolles, A., Mensch, A., Bamford, C., Chaplot, D.S., Casas, D. de las, Bressand, F., Lengyel, G., Lample, G., Saulnier, L., Lavaud, L.R., Lachaux, M.-A., Stock, P., Scao, T. Le, Lavril, T., Wang, T., Lacroix, T. and Sayed, W. El, (2023b) Mistral 7B.

Ko, B. and Choi, H.J., (2020a) Paraphrase bidirectional transformer with multi-task learning. In: *Proceedings - 2020 IEEE International Conference on Big Data and Smart Computing, BigComp 2020*. Institute of Electrical and Electronics Engineers Inc., pp.217–220.

Ko, B. and Choi, H.J., (2020b) Paraphrase bidirectional transformer with multi-task learning. In: *Proceedings - 2020 IEEE International Conference on Big Data and Smart Computing, BigComp 2020*. Institute of Electrical and Electronics Engineers Inc., pp.217–220.

Kubal, D.R. and Nimkar, A. V, (2018) *A Hybrid Deep Learning Architecture for Paraphrase Identification*.

Kuksa, V. and Polyakov, M., (2024) Developing and Applying a Neural Network System for Text Plagiarism Detection in Higher Education. In: *Proceedings - 2024 4th International Conference on Technology Enhanced Learning in Higher Education, TELE 2024*. Institute of Electrical and Electronics Engineers Inc., pp.412–416.

Kumar, N.V.A. and Mehrotra, S., (2022) A Comparative Analysis of word embedding techniques and text similarity Measures. In: *Proceedings of 5th International Conference on*

Contemporary Computing and Informatics, IC3I 2022. Institute of Electrical and Electronics Engineers Inc., pp.1581–1585.

Latina, J. V., Cabalsi, G.M., Sanchez, J.R., Vallejo, E.D.M., Centeno, C.J. and Garcia, E.A., (2024) Utilization of NLP Techniques in Plagiarism Detection System through Semantic Analysis using Word2Vec and BERT. In: *Proceedings - 2024 International Conference on Expert Clouds and Applications, ICOECA 2024*. Institute of Electrical and Electronics Engineers Inc., pp.347–352.

Le, Q. V. and Mikolov, T., (2014) Distributed Representations of Sentences and Documents. [online] Available at: http://arxiv.org/abs/1405.4053.

Li, X., Zeng, F. and Yao, C., (2020a) A Semi-Supervised Paraphrase Identification Model Based on Multi-Granularity Interaction Reasoning. *IEEE Access*, 8, pp.60790–60800.

Li, Y., Liu, D. and Liu, G., (2020b) Paraphrase Identification Based on ResNeXt. In: *2020 IEEE 3rd International Conference on Automation, Electronics and Electrical Engineering, AUTEEE 2020*. Institute of Electrical and Electronics Engineers Inc., pp.408–412.

Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., Levy, O., Lewis, M., Zettlemoyer, L. and Stoyanov, V., (2019) RoBERTa: A Robustly Optimized BERT Pretraining Approach. [online] Available at: http://arxiv.org/abs/1907.11692.

Llama Team, A., (2024) The Llama 3 Herd of Models.

Marelli, M., Menini, S., Baroni, M., Bentivogli, L., Bernardi, R. and Zamparelli, R., (2014) *A SICK cure for the evaluation of compositional distributional semantic models*. [online] Available at: http://www.cs.york.ac.uk/semeval-2012/.

Marelli, M., Menini, S., Baroni, M., Bentivogli, L., Bernardi, R. and Zamparelli, R., (n.d.) *A SICK cure for the evaluation of compositional distributional semantic models*.

Mhatre, S., Satre, S., Hajare, M., Hire, A., Itankar, A. and Patil, S., (2023) Text Comparison Based on Semantic Similarity. In: *2023 3rd International Conference on Intelligent Technologies, CONIT 2023*. Institute of Electrical and Electronics Engineers Inc.

Mikolov, T., Chen, K., Corrado, G. and Dean, J., (2013) Efficient Estimation of Word Representations in Vector Space. [online] Available at: http://arxiv.org/abs/1301.3781.

Mohamed, M. and Oussalah, M., (2020) A hybrid approach for paraphrase identification based on knowledge-enriched semantic heuristics. *Language Resources and Evaluation*, 542, pp.457–485.

Omi, A.M., Hossain, M., Islam, M.N. and Mittra, T., (2021) Multiple Authors Identification from Source Code Using Deep Learning Model. In: *Proceedings of International Conference* *on Electronics, Communications and Information Technology, ICECIT 2021*. Institute of Electrical and Electronics Engineers Inc.

Palivela, H., (2021) Optimization of paraphrase generation and identification using language models in natural language processing. *International Journal of Information Management Data Insights*, 12, p.100025.

Patil, R., Boit, S., Gudivada, V. and Nandigam, J., (2023) A Survey of Text Representation and Embedding Techniques in NLP. *IEEE Access*, 11, pp.36120–36146.

Peinelt, N., Nguyen, D. and Liakata, M., (2020) *tBERT: Topic Models and BERT Joining Forces for Semantic Similarity Detection*.

Pennington, J., Socher, R. and Manning, C.D., (2014) *GloVe: Global Vectors for Word Representation*. [online] Available at: http://nlp.

Qaiser, S. and Ali, R., (2018) Text Mining: Use of TF-IDF to Examine the Relevance of Words to Documents. *International Journal of Computer Applications*, 1811, pp.25–29.

Rosu, R., Stoica, A.S., Popescu, P.S. and Mihaescu, M.C., (2020) NLP based Deep Learning Approach for Plagiarism Detection. *International Joural of User-System Interaction*, 131, pp.48–60.

Saeed, A.A.M. and Taqa, A.Y., (2022a) A proposed approach for plagiarism detection in Article documents. *SinkrOn*, 72, pp.568–578.

Saeed, A.A.M. and Taqa, A.Y., (2022b) Textual Plagiarism Detection Using Embedding Models and Siamese LSTM. In: *2022 International Conference for Natural and Applied Sciences, ICNAS 2022*. Institute of Electrical and Electronics Engineers Inc., pp.95–100.

Saqaabi, A.A.L., Akrida, E., Cristea, A. and Stewart, C., (2022) A Paraphrase Identification Approach in Paragraph Length Texts. In: *IEEE International Conference on Data Mining Workshops, ICDMW*. IEEE Computer Society, pp.358–367.

Setha, I. and Aliane, H., (2022) Enhancing automatic plagiarism detection: Using Doc2vec model. In: *ICAASE 2022 - 5th Edition of the International Conference on Advanced Aspects of Software Engineering, Proceedings*. Institute of Electrical and Electronics Engineers Inc.

Sharma, L., Graesser, L., Nangia, N. and Evci, U., (2019) *Natural Language Understanding with the Quora Question Pairs Dataset*. [online] Available at: https://www.kaggle.com/c/quora-question-pairs.

Sharma, L., Graesser, L., Nangia, N. and Evci, U., (n.d.) *Natural Language Understanding with the Quora Question Pairs Dataset*.

Thakur, N., Reimers, N., Daxenberger, J. and Gurevych, I., (2021) Augmented SBERT: Data Augmentation Method for Improving Bi-Encoders for Pairwise Sentence Scoring Tasks. *NAACL-HLT 2021 - 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Proceedings of the Conference*, 2, pp.296–310.

Vasuteja, A., Reddy, A.V. and Pravin, A., (2024) Beyond Copy Paste: Plagiarism Detection using Machine Learning. In: *7th International Conference on Inventive Computation Technologies, ICICT 2024*. Institute of Electrical and Electronics Engineers Inc., pp.245–251.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, L. and Polosukhin, I., (2017a) Attention Is All You Need. [online] Available at: http://arxiv.org/abs/1706.03762.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, L. and Polosukhin, I., (2017b) Attention Is All You Need.

Wang, M., Xie, P., Du, Y. and Hu, X., (2023) T5-Based Model for Abstractive Summarization: A Semi-Supervised Learning Approach with Consistency Loss Functions. *Applied Sciences (Switzerland)*, 1312.

Wang, X., Li, C., Zheng, Z. and Xu, B., (2018) *Paraphrase Recognition via Combination of Neural Classifier and Keywords*. IEEE.

Warner, B., Chaffin, A., Clavié, B., Weller, O., Hallström, O., Taghadouini, S., Gallagher, A., Biswas, R., Ladhak, F., Aarsen, T., Cooper, N., Adams, G., Howard, J. and Poli, I., (2024) Smarter, Better, Faster, Longer: A Modern Bidirectional Encoder for Fast, Memory Efficient, and Long Context Finetuning and Inference. [online] Available at: http://arxiv.org/abs/2412.13663.

Yenduri, G., M, R., G, C.S., Y, S., Srivastava, G., Maddikunta, P.K.R., G, D.R., Jhaveri, R.H., B, P., Wang, W., Vasilakos, A. V. and Gadekallu, T.R., (2023) Generative Pre-trained Transformer: A Comprehensive Review on Enabling Technologies, Potential Applications, Emerging Challenges, and Future Directions. [online] Available at: http://arxiv.org/abs/2305.10435.

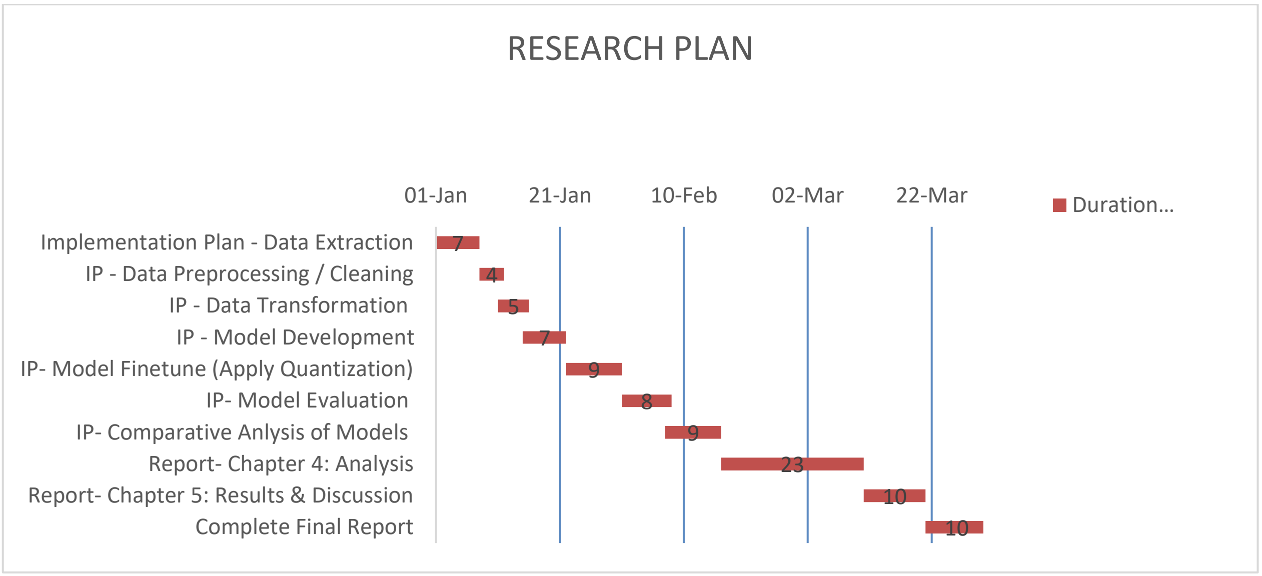
Zhang, J., Xue, S., Li, J.L. and She, J., (2022) Automated Plagiarism Detection Model Based On Deep Siamese Network. In: *Proceedings of 2022 8th IEEE International Conference on Cloud Computing and Intelligence Systems, CCIS 2022*. Institute of Electrical and Electronics Engineers Inc., pp.298–302.

Zhang, X., Rong, W., Liu, J., Tian, C. and Xiong, Z., (2017) *Convolution Neural Network Based Syntactic and Semantic Aware Paraphrase Identification*. IEEE.

Zhang, Y., Baldridge, J. and He, L., (2019a) PAWS: Paraphrase Adversaries from Word Scrambling. [online] Available at: http://arxiv.org/abs/1904.01130.

Zhang, Y., Baldridge, J. and He, L., (2019b) PAWS: Paraphrase Adversaries from Word Scrambling.

APPENDEX A: RESEARCH PLAN



In the first phase, the research will focus on selecting datasets including the MRPC dataset with moderate lexical overlap, the QQP dataset for question-answering scenarios with lesser lexical overlap, the PAWS datasets with high lexical overlap, and less explore other relevant data or resources. In the data cleaning process, noisy data and irrelevant information will be removed. If necessary, missing values will be imposed. For data preprocessing, the initial step will be calculating sentence lengths to understand the text structure. Tokenization will then be applied to convert text into tokens. Then an embedding layer will be used to map these tokens to numerical representations.

In the second phase, the research will focus on applying decoder-based LLMs such as Llama and Mistral while selecting a suitable pre-trained LLM for the task. While encoder-based models like BERT and T5 have been widely used by researchers for similar tasks, this study aims to explore the potential of decoder-based models to advance paraphrase identification. Llama has released multiple versions of LLMs with varying configurations, such as Llama 3.2 models with 1B and 3B parameters, 11B and 90B parameters, and Llama 3.1 models with 405B, 70B, and 8B parameters. The research will begin with lightweight and cost-efficient models like Llama 3.2 1B and 3B parameters to evaluate their performance. If the performance is found inadequate for the requirements, larger models will be explored to achieve better accuracy and results. Llama 3.2 (1B and 3B) are already quantized models, making them resource efficient.

However, if further optimization is feasible, additional quantization techniques will be applied to minimize resource allocation while maintaining or improving performance.
In the third phase, the research will focus on model evaluation. Evaluation metrics are used to measure the quality of the model. Since the datasets include a label column with binary values, a confused matrix will be used for evaluation. Accuracy, precision, recall are the main key metrics which will focus on this research. The PAWS dataset is balanced, so it will use a confusion matrix. However, if other datasets are found to be imbalanced, then we will use AUC-ROC. In the fourth phase, the research will focus on comparative analysis of encoder-based and decoder-based models. This analysis will evaluate the strength and weakness of the model type, aiming to identify which architecture is best suited for paraphrase identification tasks. The analysis will also consider resource usage, including memory and processing power, especially when employing quantization techniques. The goal is to provide an understanding of how different model architectures and optimizations influence both performance and practicality.

APPENDEX B: RESEARCH PROPOSAL

PARAPHRASE IDENTIFICATION USING LLM MODELS AND COMPARATIVE ANALYSIS WITH FINE TUNED LLM MODEL AND BASE TRANSFORMER MODEL

RAHUL NANDI

Research Proposal

NOVEMBER 2024

Abstract

Paraphrase identification is a critical task in natural language processing (NLP) that determines whether two sentences convey the same meaning. This research focuses on leveraging large language models (LLMs) and resource-efficient optimization techniques to improve paraphrase identification across benchmark datasets, including MRPC, Quora Question Pairs (QQP), and Paraphrase Adversaries from Word Scrambling (PAWS). A particular emphasis is placed on decoder-based LLMs, such as LLaMA and Mistral, and the application of quantization techniques (e.g., 1-bit and 4-bit) to optimize resource usage without compromising performance. This research contributes to bridging gaps in decoder-based model utilization and provides practical insights into the effectiveness of quantization techniques for paraphrase identification.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF ABBREVIATIONS

PI: Paraphrase Identification

NLP: Natural Language Processing

TF-IDF: Term Frequency Inverse Document Frequency

BERT: Bidirectional Encoder Representations from Transformers

MRPC: Microsoft Research Paraphrase Corpus

QQP: Quora Question Pairs

PAWS: Paraphrase Adversaries from Word Scrambling

SICK: Sentences Involving Compositional Knowledge

LLM: Large Language Models

MT-DNN: Multi-Task Deep Neural Network

T5: Text-To-Text Transfer Transformer

Elmo: Embedding from Language Models

LDA: Latent Dirichlet Allocation

NLM: Neural Language Model

GPT: Generative Pre-Trained Transformers

RELU: Rectified Linear Unit

MoE: Mixture of Experts

PEFT: Parameter Efficient Fine-Tuning

LoRA: Low-Rank Adaptation

QLoRA: Quantized Low-Rank Adaptation

ROC: Receiver Operating Characteristics

AUC: Area Under Curve

1. Background

Paraphrase identification (PI) is beneficial for several NLP task. Like plagiarism detection, questions-answering performance improvement, text summarization, and machine translation etc., it is used to figure out whether two text sequences or documents convey the same meaning or not, In the research field, paraphrase identification can be used to evaluate plagiarism detection. In the education field, whether the answer is semantically equivalent to a reference answer or not? In summarization, is used to eliminate redundant information. Text similarity is one of the ways to identify whether two pair of sentence or sequence are identical or not, if identical then it call as paraphrase, otherwise non-paraphrase.

To identify paraphrases or text similarity effectively, two key approaches are commonly used: lexical similarity and semantic similarity.

Lexical similarity: The lexical similarity focuses on how similar two text sequences are in terms of their surface-level features, such as word overlap, structure, or syntax. For example, phrases are “the cat ate the mouse” and “the mouse ate the cat food”. by just looking at the words? On the surface, if consider only word level similarity, these two phrases appear very similar as 3 of the 4 unique words are an exact overlap.

Semantic Similarity: The semantic similarity focuses on how similar two sequence are in terms of their meaning rather than surface level. Methods of calculating semantic similarity: Cosine similarity, jaccard similarity, word embedding-based, Siamese Networks, attention-based models.

Types of Semantic Similarity:

Knowledge-Based Similarity: Knowledge-based similarity measures how similar two words are based on their meaning, using information from semantic networks. One of the most widely used semantic networks is WordNet, a large database of English words. In WordNet, words are categorized as nouns, verbs, adjectives, or adverbs and grouped into synonym sets, with each group representing a specific concept.

Statistical-Based Similarity: These methods calculate similarity using statistical measures applied to text features or embeddings. Example method Cosine similarity between vectorized text representations (e.g., TF-IDF, word embeddings).

String-Based Similarity: String-based similarity measures compare the text itself rather than the meaning of the words or sentences. Examples include methods like edit distance and Jaccard similarity.

Language Model-Based Similarity: Language model-based similarity measures use pre-trained language models to understand the meaning of the text. They compare the encoded representations of the text to determine similarity. Examples include models like BERT and Transformers.

For paraphrase identification, several tech companies like Google, Microsoft, etc., released large-scale datasets. Most of the research happened using these datasets. MRPC (Dolan and Brockett, n.d.) released Microsoft. Quora Question Pairs (QQP) (Sharma et al., n.d.) released by Quora, Paraphrase Adversaries from Word Scrambling (PAWS) (Zhang et al., 2019) released by Google. Each of these datasets includes 3 columns: sentence1, sentence2 and label column. Label column indicates whether 2 sentences are paraphrased or not. Based on that, everyone is applying classification matrices to evaluate the result. So precision, recall, F1score and other metrics need to be applied to evaluate the model performance.

2. Related Work

Research gap are existing research primarily focused either traditional ML models or encoder based large language models (LLM) like Bidirectional Encoder Representations from Transformers (BERT), T5 models. Limited exploration of decoder-based LLMs like Llama, Mistral etc. also not focused on resource-efficient deployment.

The MRPC (Dolan and Brockett, n.d.) comprises 5,801 sentence pairs, with 67% of them being judged semantically equivalent based on human evaluation. Another widely used dataset is the QQP (Quora Question Pairs), which contains questions collected from the Quora platform. Identifying semantically equivalent questions is a crucial task for any question-answering system, including platforms like Quora and Stack Overflow. However, QQP exhibits relatively low lexical overlap between sentence pairs. To address this limitation, (Zhang et al., 2019b) introduced the PAWS dataset (Paraphrase Adversaries from Word Scrambling), which includes 108,463 sentence pairs with higher lexical overlap. Sentences Involving Compositional Knowledge (SICK) (Marelli et al., n.d.) includes a large number of pairs of sentence that have lexical, syntactic and semantic phenomena.

Nicholas Gahman and Vinayak Elangovan have compared 4 types of algorithms: BERT + Cosine Similarity, Multi-Task Deep Neural Network (MT-DNN), XLNet, and Lin Measure + String Similarity, to find out the document databases on multiple dataset MRPC, AFS, SICK-E and SICK-R datasets. They first applied stop word then word embedding technique (BERT, BoW, Sent2Vec), then using cosine, Euclidian and Manhattan distance, they found the similarity. They found MT-DNN is the best model for document similarity. (Gahman and Elangovan, 2023). Abu Bakar Siddiqur Rahman, along with other researchers, they proposed 2 lightweight models for paraphrase identification, Linear regression and multilayer perceptron. For mlp they used 3 hidden layers with learning_rate_init=0.01, random_rate=5, activation='relu', and. Also, they used pre-trained models for pair similarity from hugging face but linear regression with noise filtering gives best result. (Bakar et al., 2022)

(Thakur et al., 2021) Nandan thakur, alone with other researchers, are working on improving bi-encoders to make them better at scoring pairs of sentences. Bi-encoders encode sentences independently but often require a lot of training data. Augmented SBERT tackles this issue by leveraging a powerful cross-encoder to "soft-label" new sentence pairs. These soft-labeled pairs are then incorporated into the training data for the bi-encoder. This approach significantly improves performance, especially for domain adaptation tasks where labeled data might be scarce. (Palivela, 2021) Hemant Palivela used an encoder based fine-tuned Text-To-Text Transfer Transformer (T5) model for paraphrase identification as well as paraphrase generation. , also used sentence embedding and semantic similarity scores, which helped to achieve paraphrase identification.

(Mohamed and Oussalah, 2020), They proposed a hybrid approach for paraphrase identification. They used 2 database wordnets and categorical variation. Firstly, they pre-processed the data like text cleaning, tokenizing, part of speech tagging, stop word removal, etc. then apply normalization, which will convert verb, adjective, and adverb categories to nouns. After that, they applied the formula and found out the similarity score after combining Wikipedia-based name entity similarity and Catvar-aided wordNet-based similarity.

(Gangadharan Veena et al., 2020), They used different word vectorization techniques for finding paraphrase, paraphrases also proposed which vectorization method is more efficient. Technique of word vectorization: Count vectorizer, Term frequency inverse document frequency (TF-IDF), Hashing vectorizer, Word2Vec, Glove, fastText, Embedding from Language Models (Elmo), and BERT. And for document similarity, they used cosine similarity. As per them, fastText is the better method for converting best to a vector and to find its similarity. (Peinelt et al., 2020), They combined topics with BERT for semantic similarity prediction, and it is called tBERT. Latent Dirichlet Allocation (LDA) is used for models. (Ko and Choi, 2020a) Proposed paraphrase-BERT. They used MRPC data to fine-tune the pre-trained BERT model, then performed a multi-task (MLT) to improve performance. Also, we compared performance between multiple models: the Neural Language Model (NLM), the ARC model, and Bi-CNN-MI.

3. Research Questions

- How well do decoder-based models (e.g., Llama, mistral) perform on paraphrase identification compared to encoder-based models (e.g., BERT)?
- Can resource usage be optimized through fine-tuning without significant performance degradation?
- What fine-tuning strategies can enhance the performance of LLMs on paraphrase identification?

4. Aim and Objectives

The main aim of this research is identifying paraphrase using fine-tuned LLM model. And comparative analysis between fine-tuned LLM model and transformers models like BERT. The identification of the paraphrase can be used for application such as plagiarism detection.

The research objectives are formulated based on the aim of the study, which are as follows:

- To identify paraphrases using the LLM model.
- To analyze the performance of existing LLM models and tried to optimize the resource usage.
- To Comparative analysis between decoder-based model likes (LlaMA, Mistral etc.) and encoder-based model likes (BERT, T5 etc.)
- To explore methods to optimize or fine tune the LLM Model.

5. Significance of the Study

Paraphrase identification (PI) is useful for several natural language processing applications across various domains. Plagiarism detection, question-answering systems, text summarization, and machine translation are the few tasks. PI is used to find out whether two text sequences convey the same meaning or not. In the educational field, to evaluate whether student response aligns semantically with the reference answer. It will improve the functionality of search engines and question-answering systems by better understanding user intent. In the research area, it will help to identify plagiarism in research documents. In summarization, it is used to cut redundant information. Sentences are paraphrased in 2 ways. Sentences are lexical overlap and semantic wise paraphrased. If sentences lexically overlap without being paraphrased, then most of the models fail to distinguish pairs, like flights from the United States to India and flights from India to the United States. This study emphasizes the critical role of understanding lexical & semantic paraphrase.

A primary contribution of this study is addressing existing research gaps in paraphrase identification. While many studies have focused on encoder-based language models (Like BERT, T5) and limited exploration on decoder-based LLM models (GPT, Llama, Mistral etc.) there are also very limited exploration into resource optimization and lightweight methods. This research will apply quantization techniques on pre-trained language models and try to optimize resource usages. A major advantage of optimizing resource usage makes the findings relevant for deploying NLP models on edge devices and other small devices which have limited computational power. This study not only the understanding of paraphrase identification techniques but also contributes to the other NLP areas like multilingual paraphrase identification, domain-specific applications etc.

6. Scope of the Study

It specifically uses datasets such as the MRPC , QQP (Quora Question Pairs), and Paraphrase Adversaries from Word Scrambling (PAWS). These datasets were selected for their diversity in lexical and semantic similarity characteristics. This study focuses on sentence-level paraphrase identification using LLMs and fine tune the LLMs and achieve the same model performance on 1.58-bit or 4-bit. The evaluation metrics used in this study are precision, recall, and F1 score, which are standard measures in classification tasks. These metrics helps to get model performance, ensuring the reliability and generalizability of the results. The study also includes a comparative analysis of encoder-based and decoder-based models, highlighting their strengths and limitations in paraphrase identification.

While the primary focus is on sentence-level paraphrase identification, the study's scope does not extend to paragraph-level or document-level paraphrase detection. The datasets used are primarily English. However, the methodologies and insights derived from this research can be extended to other languages and domains in future studies. Open-source decoder based LLMs going to use for this study.

7. Research Methodology

7.1 Introduction

In the field of Natural Language Processing (NLP), paraphrase detection plays a critical role in application such as plagiarism detection, question answering, information retrieval, and text summarization etc. This research aims to improve paraphrase detection by leveraging large language models (LLMs) and comparative analysis with transformers-based models like Bidirectional Encoder Representations from Transformers (BERT).

7.2 Dataset Description

In this proposal we are using PAWS dataset (Zhang et al., 2019b). It has 108463 well-formed paraphrase and non-paraphrase pairs with high lexical overlap. This dataset has pairs generated from Wikipedia pages. There are 3 subsets of data that is labeled_final, labeled_swap and unlabeled_final, which contains 4 columns id, sentence1, sentence2 and label with 3 subsets of data. The MRPC (Dolan and Brockett, n.d.) consists of 5,801 sentence pairs, with 67% deemed semantically equivalent based on human evaluation. Another prominent dataset is the Quora Question Pairs (QQP) (Sharma et al., n.d.), which includes questions sourced from the Quora platform.

7.3 Data Pre-processing

As part of data pre-processing, I have to break down the text into individual token.

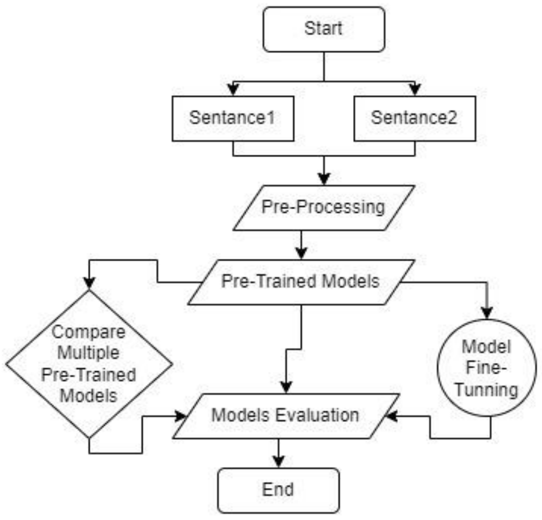


Fig1: Overview of the Proposal Method

7.4 Transformation

I will use embedding technique to convert the text into numerical representations. Word embedding are numerical representations of words that show semantic similarities and correlations depending on how often they appear in a dataset.

7.5 Models

All the LLM models based on Transformers. So will discuss Transformers then will look at one of the open LLM models.

Transformers is a type of neural network architecture. Transformers was developed by Google in 2017 (Vaswani et al., 2017b). It is mainly developed to solve the problem of sequence transduction, or neural machine translation and some other tasks. It will learn the context of sequential data and generate new data out of it. Transformers architecture has two main blocks: Encoder and Decoder.

Encoder: An encoder is used for processing the input sequence. It has a Multi-Head Attention mechanism and a fully connected Feed-Forward network, plus layer normalization for the output of each sub-layers. There are also residual connections around the two sub-layers.

Decoder: A decoder is used for generating the output sequence. It follows a similar structure, other than attention layers. It includes two types of attention sub-layers: masked multi-head attention and encoder-decoder attention.

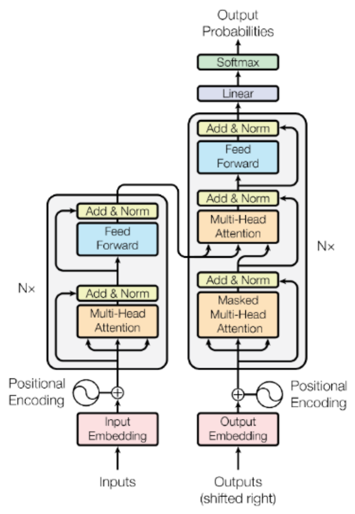


Fig. 2 : The Transformer – Model Architecture (Vaswani et al., 2017b)

Encoder each layer step by step:

Input Embedding: Input embedding is the first step of both encoder and decoder. It is responsible for converting the input text into numerical vectors of d_model dimensions. To prevent input embedding become extremely small by normalizing them, multiple them by the $\sqrt{d\ model}$.

Positional Encoding: Positional encoding is responsible for preserving sequence order, maintaining contextual information. This will help model to understand the sequential nature of data (like sentence). For positional encoding They use sine and cosine functions with different frequencies for positional encoding.:

$PE_{(pos,2i)} = \sin(pos/10000^{2i/dmodel})$

$PE_{(pos,2i+1)} = \cos(pos/10000^{2i/dmodel})$

Where:

Pos = Position in sequence

i = Dimension of embedding (split into even and odd indices)

d = Dimension of the encoding

Add & Norm (Additional and Normalization): When we look at the encoder and decoder blocks, we see several normalization layers called add & norm. These layers are crucial for stabilizing and enhancing the learning process. It consists of two essential components: a residual connection (Add) and layer normalization (norm).

Residual Connection (add): Each sub-layer, including the self-attention and Feed Forward blocks, adds its output to its input before passing it to the Add & Norm layer. This process is known as skip connection. This approach allows the model to keep the original information from previous layers, helping gradients flow more smoothly and mitigating the vanishing gradient problem.

Normalization: Layer normalization formula is used. This process results in a normalized output with a mean 0 and a standard deviation 1.

$$\text{Norm}(X) = \frac{x-\mu}{\sigma+\varepsilon}$$

Where:

x: input data

μ: mean of the features

σ: Standard deviation

ε : epsilon (to avoid any divisions by zero)

Attention: Attention is the most crucial component of the Transformers. It is responsible for helping the model to understand complex relationship and patterns in the sentence. It assigns weights to each word based on the relevance and influence, works by seeing how similar and important each word is to all of the words in a sentence, including itself. This enables the model to capture long-range dependencies.

Self-Attention Mechanism: Self-attention, also known as scaled dot-product attention. The self-attention mechanism creates three vectors: Query (Q), Key (K), and Value (V). These vectors are learned during training.

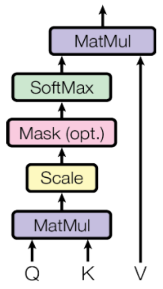


Fig. 3: Scaled Dot-Product Attention (Vaswani et al., 2017b)

Attention Function:

$$\text{Attention(Q,K,V)} = \text{Softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$

Query (Q): Represents the word we are focusing on to calculate attention scores. Key (K): Represents all the other words in the sentence for comparison. Value (V): Represents the information from all the words, which will be used to create a weighted sum.

Attention Scores: The model finds attention scores by multiplying the Query vector of the current word with the Key vectors of all the words in the input. These scores indicate how related each word is to the current word.

Scale: The dot products are reduced by dividing them by the square root of the size of the Key vectors.

SoftMax: The scaled dot products go through a SoftMax function to turn them into a probability distribution. This function makes sure the scores are between 0 and 1 and add up to 1.
Weighted Sum: The attention weights from SoftMax are used to create a weighted sum of the value vectors of all words. This sum represents the new version of the current word, including information from other words based on how important they are.

Multi-Head Attention: Instead of using just one attention mechanism, the model uses multiple heads, each with its own set of Query, Key, and Value vectors. This allows each head to focus on different parts of the input, capturing various aspects of word relationships. As per paper they use 8 heads or parallel attention layers.

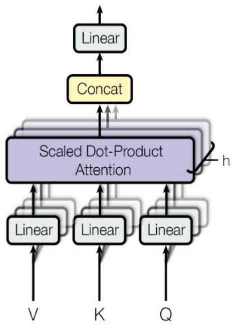


Fig. 4: Scaled Dot-Product Attention (Vaswani et al., 2017b)

$$\text{MultiHead}(Q,K,V) = \text{concat}(\text{head}_1,\ldots.\text{head}_h)W^0$$

Where:

$\text{head}_i = \text{Attention}\ (QW_i^Q, KW_i^K, VW_i^V\ )$

$W_i^Q \in \mathbb{R}^{d_{model} \times d_k}$

$W_i^K \in \mathbb{R}^{d_{model} \times d_k}$

$W_i^V \in \mathbb{R}^{d_{model} \times d_v}$

$W_i^0 \in \mathbb{R}^{hd_v \times d_{model}}$

Feed-Forward: This is fully connected feed-forward network layer. They applied two linear transformations with a ReLU activation in between. The dimensionality of input and output is $d_{model} = 512$, neurons (d_{ff})are 2048.

$$\text{FFN}(x) = \max(0, xW_1 + b_1)W_2 + b_2$$

Where:

W_1 and W_2 are the weights.

b_1 and b_2 are the biases.

Decoder each layer:

As noted above, the decoder follows the same structure as the encoder, except attention layers.

Masked Multi-Head Self-Attention: This type of self-attention hides certain tokens in the input sequence or future tokens. This ensures that the decoder can only use information from past tokens and the current token to predict the next word in the output sequence.

Masked Attention Function:

$$\text{Masked Attention(Q,K,V)} = \text{Softmax}\left(\frac{QK^T}{\sqrt{d_k}} + mask\right)V$$

Where:

mask: large negative number (e.g., -inf)

Encoder-Decoder Attention: In this attention layer, the keys and values come from the decoder's output, while the queries are based on the decoder's current representation.

7.5.1 Llama 2 and Llama3 Architecture (Llama Team, 2024)

Llama model is released by meta-AI. This is family of autoregressive large language models.

Exploring Llama: Llama builds upon and significantly modifieds the plain transformer.

Rotary Positional Encoding: This replaces the standard positional encoding, providing a more effective way to maintain the relative positional information between tokens in a sequence, enhancing the model's ability to understand and generate more coherent texts.

Single Layer Architecture: Unlike the encoder-decoder structure of Plain Transformers, Llama optimizes to a more streamlined, single-layer approach, improving efficiency without sacrificing performance.

RMS Norm: Replacing LayerNorm, RMS Norm is employed for normalization within the network, offering improved training stability and efficiency.

Grouped Multi-Query Attention: An evolution from self-attention, this mechanism allows the model to process multiple queries in groups, enhancing the attention mechanism's efficiency and capacity to handle complex patterns.

KV Cache: A novel addition does not present in Plain Transformers, enabling the model to cache previous computations, significantly speeding up the processing of sequences by reducing redundancy.

SwiGLU Activation in Feedforward Networks: Instead of RELU activation, Llama utilizes SwiGLU, offering a more powerful and efficient activation mechanism that contributes to the model's overall improved performance.

7.5.2 Mistral Architecture (Jiang et al., 2023b)

Mistral, while building upon Llama, introduces its own set of advancements to push the envelope further:

Sliding Window Attention: This new attention mechanism allows Mistral to focus on subsets of the input data more efficiently, improving the model's ability to manage and interpret large sequences with greater precision.

Rolling Buffer KV Cache: An enhancement over Llama's KV cache, the rolling buffer mechanism optimizes memory usage further, enabling even more efficient handling of long sequences.

Mixture of Experts (MoE) in Feedforward Networks: Building on the SwiGLU-based feedforward of Llama, Mistral incorporates MoE to dynamically route different parts of the input through different specialized 'expert' networks, significantly boosting the model's capacity and efficiency.

7.6 LLM Models Fine-tuning Techniques

Parameter Efficient Fine-Tuning (PEFT): PEFT is a transfer learning approach that tackles the challenges of fully fine-tuning large language models (LLMs). It involves keeping all the pre-trained model’s original parameters unchanged and adding new parameters that can be adjusted during fine-tuning. Adapters are a type of submodule added to pre-trained models to adjust their hidden representations during fine-tuning. By placing adapters after the multi-head attention and feed-forward layers in the transformer architecture, only the adapter parameters are updated during fine-tuning, while the rest of the model remains frozen. There are different methods for PEFT, with Low-Rank Adaptation (LoRA) and QLoRA being the most popular and effective.

Benefits of PEFT: Decreased computational and storage costs, Overcoming catastrophic forgetting, Portability, Sustainability, Lower storage requirements etc.

7.6.1 Low-Rank Adaptation (LoRA) (Hu et al., 2021):

Low-Rank Adaption (LoRA) primarily developed by Microsoft. LoRA can reduce the number of trainable parameters by 10000 times and the GPU memory requirement by 3 times.

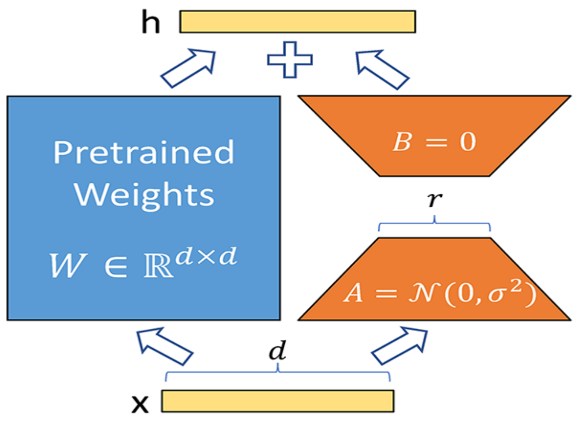

h
Pretrained Weights
$W \in \mathbb{R}^{d \times d}$
$B = 0$
r
$A = \mathcal{N}(0, \sigma^2)$
d
x

Fig. 5: LoRA Reparameterization (Hu et al., 2021)

LoRA decomposes the weight matrix(W_0) into the two smaller matrices A and B such that $\Delta W=BA$, where A and B have much lower ranks than W_0. During finetuning, only the matrices A and B are updated while W_0 remains fixed.

$$h = W_0 + \Delta W = W0 + BA$$

Where:

W_0: Actual pre-trained Weights

$W_0 \in \mathbb{R}^{d \times k}$

A & B: Two smaller matrix

$A \in \mathbb{R}^{d \times r}, B \in \mathbb{R}^{r \times k}$

r: hyperparameter (values1,2,4,8 or 64)

$r << \min(d, k)$

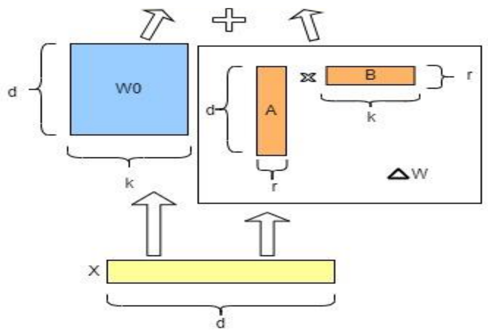


Fig. 6: LoRA Adaptation

7.6.2 QLoRA (Quantized Low-Rank Adaptation)

QLoRA is an extension of LoRA that makes the method even more efficient. Quantization is a process of reducing the precision of the numbers used to represent model weights. For instance, instead of storing weights as 32-bit floating points, you might store them using just 4 bits. This significant reduction in data size means that the model requires less memory, and computations can be executed more quickly and with less energy.

7.7 Evaluation metrics

Train test split is not needed as data has already split. Will use standard binary classification metrics to evaluate our models:

Confusion Matrix: A confusion matrix displays the counts of correct and incorrect predictions made by a classification model, comparing them to the actual outcomes in the data.

Accuracy: Accuracy measures the total number of correct classifications divided by the total number of cases.

Recall/True Positive Rate/ Sensitivity: Recall/Sensitivity tells us out of the actual positive values, how many of them have predicted as positive.

Precision: Precision tells us out of all the predicted positive values, how many of them are actual positive.

Specificity: Specificity tells us out of all the actually negative, how many of them are correctly predicted negative.

F1 Score: F1 Score is a single metric that is a harmonic mean of precision and recall.

ROC (Receiver Operating Characteristics) AUC (Area Under Curve) Score: It is a graphical representation of the model performance at various threshold levels. It shows the trade-off between the true-positive rate and false positive rate.

8. Required Resources

Hardware Requirement: Multi-GPU Setup, Multi-Core CPU, RAM: 32-128GB, Storage-250GB-1TB SSD, any cloud setup will be required.

Software Requirement:

Language: Python version:3.9 or later
3rd party Libraries:

- PyTorch: For Deep leaning model implementation,
- Hugging Face Transformers: For load the open-source LLM Models,
- Datasets: For loading the dataset,
- PEFT: For fine tuning the model,
- Bitsandbytes: For apply quantization,
- Accelerate: For Multi-GPU or TPU Setup,
- Pandas, NumPy: Data Processing,
- Scikit-Learn: For model Evaluation metrics (Confusion Matrix) and others,
- Matplotlib, Seaborn: Data Visualization,
- Trl: For apply Reinforcement Learning

9. Research Plan

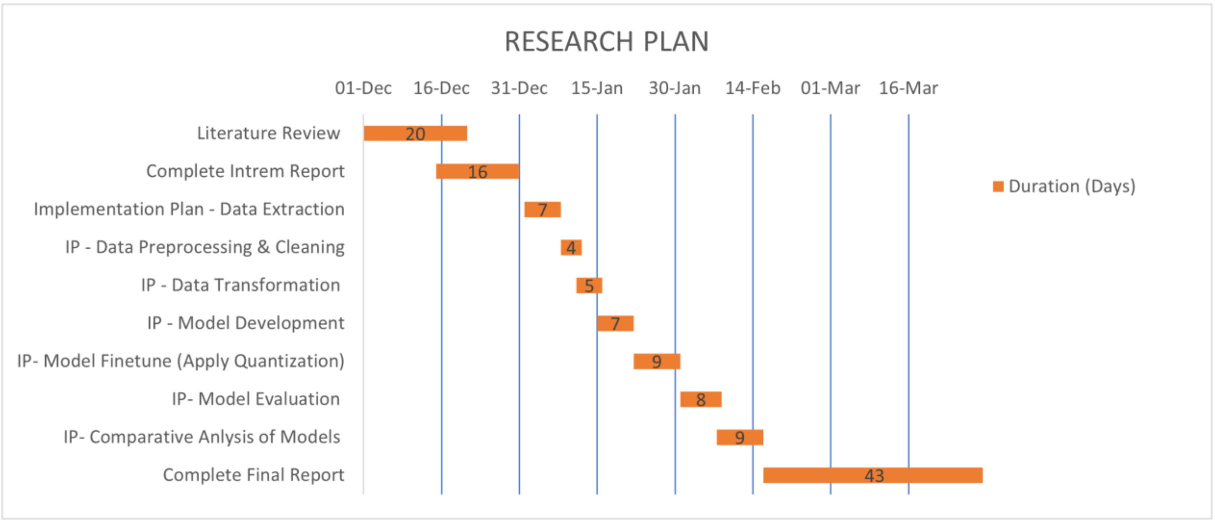


Fig. 7: Gantt Chart for Research Plan

In the first phase, the research will focus on selecting datasets including the MRPC dataset with moderate lexical overlap, the QQP dataset for question-answering scenarios with lesser lexical overlap, the PAWS datasets with high lexical overlap, and less explore other relevant data or resources. In the data cleaning process, noisy data and irrelevant information will be removed. If necessary, missing values will be imposed. For data preprocessing, the initial step will be calculating sentence lengths to understand the text structure. Tokenization will then be applied to convert text into tokens. Then an embedding layer will be used to map these tokens to numerical representations.

In the second phase, the research will focus on applying decoder-based large language models (LLMs) such as Llama and Mistral while selecting a suitable pre-trained LLM for the task. While encoder-based models like BERT and T5 have been widely used by researchers for similar tasks, this study aims to explore the potential of decoder-based models to advance paraphrase identification. Llama has released multiple versions of LLMs with varying configurations, such as Llama 3.2 models with 1B and 3B parameters, 11B and 90B parameters, and Llama 3.1 models with 405B, 70B, and 8B parameters. The research will begin with lightweight and cost-efficient models like Llama 3.2 1B and 3B parameters to evaluate their performance. If the performance is found inadequate for the requirements, larger models will be explored to achieve better accuracy and results. Llama 3.2 (1B and 3B) are already quantized models, making them resource efficient. However, if further optimization is feasible, additional quantization techniques will be applied to minimize resource allocation while maintaining or improving performance.

In the third phase, the research will focus on model evaluation. Evaluation metrics are used to measure the quality of the model. Since the datasets include a label column with binary values, a confused matrix will be used for evaluation. Accuracy, precision, recall are the main key metrics which will focus on this research. The PAWS dataset is balanced, so it will use a confusion matrix. However, if other datasets are found to be imbalanced, then we will use AUC-ROC. In the fourth phase, the research will focus on comparative analysis of encoder-based and decoder-based models. This analysis will evaluate the strength and weakness of the model type, aiming to identify which architecture is best suited for paraphrase identification tasks. The analysis will also consider resource usage, including memory and processing power, especially when employing quantization techniques. The goal is to provide an understanding of how different model architectures and optimizations influence both performance and practicality.

References

Al-Jibory, F.K. and Al-Tamimi, M.S.H., (2021) *Hybrid System for Plagiarism Detection on A Scientific Paper*. [online] *Turkish Journal of Computer and Mathematics Education*, Available at: www.gutenberg.org.

Aziz, A.A., Djamal, E.C. and Ilyas, R., (2019) *Paraphrase Detection Using Manhattan's Recurrent Neural Networks and Long Short-Term Memory*.

Aziz, A.A., Djamal, E.C. and Ilyas, R., (2024) *Siamese Similarity Between Two Sentences Using Manhattan's Recurrent Neural Networks*.

Bakar, A., Rahman, S., Thang Ta, H., Najjar, L. and Gelbukh, A., (2022) *Paraphrase Identification: Lightweight Effective Methods Based Features from Pre-trained Models*.

Bojanowski, P., Grave, E., Joulin, A. and Mikolov, T., (2016) Enriching Word Vectors with Subword Information. [online] Available at: http://arxiv.org/abs/1607.04606.

Chavan Hiten, Taufik Mohd., kadave Rutuja and Chandra Nikita, (2021) Plagiarism Detector Using Machine Learning.

Dolan, W.B. and Brockett, C., (2005) *Automatically Constructing a Corpus of Sentential Paraphrases*. [online] Available at: http://www.cogsci.princeton.edu/~wn/.

Dolan, W.B. and Brockett, C., (n.d.) *Automatically Constructing a Corpus of Sentential Paraphrases*.

Eppa, A. and Murali, A.H., (2021) Machine Learning Techniques for Multisource Plagiarism Detection. In: *CSITSS 2021 - 2021 5th International Conference on Computational Systems and Information Technology for Sustainable Solutions, Proceedings*. Institute of Electrical and Electronics Engineers Inc.

Fan, A., Wang, S. and Wang, Y., (2024) Legal Document Similarity Matching Based on Ensemble Learning. *IEEE Access*, 12, pp.33910–33922.

Gahman, N. and Elangovan, V., (2023) A Comparison of Document Similarity Algorithms. *International Journal of Artificial Intelligence & Applications*, 142, pp.41–50.

Gangadharan Veena, A Athira T, L Amritha and Gupta Deepa, (2020) *Paraphrase Detection Using Deep Neural Network Based Word Embedding Techniques*. IEEE.

Gontumukkala, S.S.T., Godavarthi, Y.S.V., Gonugunta, B.R.R.T., Gupta, D. and Palaniswamy, S., (2022) Quora Question Pairs Identification and Insincere Questions Classification. In: *2022 13th International Conference on Computing Communication and Networking Technologies, ICCCNT 2022*. Institute of Electrical and Electronics Engineers Inc.

Hu, E.J., Shen, Y., Wallis, P., Allen-Zhu, Z., Li, Y., Wang, S., Wang, L. and Chen, W., (2021) LoRA: Low-Rank Adaptation of Large Language Models.

Hunt, E., Janamsetty, R., Kinares, C., Koh, C., Sanchez, A., Zhan, F., Ozdemir, M., Waseem, S., Yolcu, O., Dahal, B., Zhan, J., Gewali, L. and Oh, P., (2019) Machine learning models for paraphrase identification and its applications on plagiarism detection. In: *Proceedings - 10th IEEE International Conference on Big Knowledge, ICBK 2019*. Institute of Electrical and Electronics Engineers Inc., pp.97–104.

Islam, A., Rahman, E., Chowdhury, A.A. and Mojumder, M.A.N., (2023) A Deep Learning Approach to Detect Plagiarism in Bengali Textual Content using Similarity Algorithms. In: *Proceedings of IEEE InC4 2023 - 2023 IEEE International Conference on Contemporary Computing and Communications*. Institute of Electrical and Electronics Engineers Inc.

Jiang, A.Q., Sablayrolles, A., Mensch, A., Bamford, C., Chaplot, D.S., Casas, D. de las, Bressand, F., Lengyel, G., Lample, G., Saulnier, L., Lavaud, L.R., Lachaux, M.-A., Stock, P., Scao, T. Le, Lavril, T., Wang, T., Lacroix, T. and Sayed, W. El, (2023a) Mistral 7B. [online] Available at: http://arxiv.org/abs/2310.06825.

Jiang, A.Q., Sablayrolles, A., Mensch, A., Bamford, C., Chaplot, D.S., Casas, D. de las, Bressand, F., Lengyel, G., Lample, G., Saulnier, L., Lavaud, L.R., Lachaux, M.-A., Stock, P., Scao, T. Le, Lavril, T., Wang, T., Lacroix, T. and Sayed, W. El, (2023b) Mistral 7B.

Ko, B. and Choi, H.J., (2020a) Paraphrase bidirectional transformer with multi-task learning. In: *Proceedings - 2020 IEEE International Conference on Big Data and Smart Computing, BigComp 2020*. Institute of Electrical and Electronics Engineers Inc., pp.217–220.

Ko, B. and Choi, H.J., (2020b) Paraphrase bidirectional transformer with multi-task learning. In: *Proceedings - 2020 IEEE International Conference on Big Data and Smart Computing, BigComp 2020*. Institute of Electrical and Electronics Engineers Inc., pp.217–220.

Kubal, D.R. and Nimkar, A. V, (2018) *A Hybrid Deep Learning Architecture for Paraphrase Identification*.

Kuksa, V. and Polyakov, M., (2024) Developing and Applying a Neural Network System for Text Plagiarism Detection in Higher Education. In: *Proceedings - 2024 4th International Conference on Technology Enhanced Learning in Higher Education, TELE 2024*. Institute of Electrical and Electronics Engineers Inc., pp.412–416.

Kumar, N.V.A. and Mehrotra, S., (2022) A Comparative Analysis of word embedding techniques and text similarity Measures. In: *Proceedings of 5th International Conference on Contemporary Computing and Informatics, IC3I 2022*. Institute of Electrical and Electronics Engineers Inc., pp.1581–1585.

Latina, J. V., Cabalsi, G.M., Sanchez, J.R., Vallejo, E.D.M., Centeno, C.J. and Garcia, E.A., (2024) Utilization of NLP Techniques in Plagiarism Detection System through Semantic Analysis using Word2Vec and BERT. In: *Proceedings - 2024 International Conference on Expert Clouds and Applications, ICOECA 2024*. Institute of Electrical and Electronics Engineers Inc., pp.347–352.

Le, Q. V. and Mikolov, T., (2014) Distributed Representations of Sentences and Documents. [online] Available at: http://arxiv.org/abs/1405.4053.

Li, X., Zeng, F. and Yao, C., (2020a) A Semi-Supervised Paraphrase Identification Model Based on Multi-Granularity Interaction Reasoning. *IEEE Access*, 8, pp.60790–60800.

Li, Y., Liu, D. and Liu, G., (2020b) Paraphrase Identification Based on ResNeXt. In: *2020 IEEE 3rd International Conference on Automation, Electronics and Electrical Engineering, AUTEEE 2020*. Institute of Electrical and Electronics Engineers Inc., pp.408–412.

Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., Levy, O., Lewis, M., Zettlemoyer, L. and Stoyanov, V., (2019) RoBERTa: A Robustly Optimized BERT Pretraining Approach. [online] Available at: http://arxiv.org/abs/1907.11692.

Llama Team, A., (2024) The Llama 3 Herd of Models.

Marelli, M., Menini, S., Baroni, M., Bentivogli, L., Bernardi, R. and Zamparelli, R., (2014) *A SICK cure for the evaluation of compositional distributional semantic models*. [online] Available at: http://www.cs.york.ac.uk/semeval-2012/.

Marelli, M., Menini, S., Baroni, M., Bentivogli, L., Bernardi, R. and Zamparelli, R., (n.d.) *A SICK cure for the evaluation of compositional distributional semantic models*.

Mhatre, S., Satre, S., Hajare, M., Hire, A., Itankar, A. and Patil, S., (2023) Text Comparison Based on Semantic Similarity. In: *2023 3rd International Conference on Intelligent Technologies, CONIT 2023*. Institute of Electrical and Electronics Engineers Inc.

Mikolov, T., Chen, K., Corrado, G. and Dean, J., (2013) Efficient Estimation of Word Representations in Vector Space. [online] Available at: http://arxiv.org/abs/1301.3781.

Mohamed, M. and Oussalah, M., (2020) A hybrid approach for paraphrase identification based on knowledge-enriched semantic heuristics. *Language Resources and Evaluation*, 542, pp.457–485.

Omi, A.M., Hossain, M., Islam, M.N. and Mittra, T., (2021) Multiple Authors Identification from Source Code Using Deep Learning Model. In: *Proceedings of International Conference on Electronics, Communications and Information Technology, ICECIT 2021*. Institute of Electrical and Electronics Engineers Inc.

Palivela, H., (2021) Optimization of paraphrase generation and identification using language models in natural language processing. *International Journal of Information Management Data Insights*, 12, p.100025.

Patil, R., Boit, S., Gudivada, V. and Nandigam, J., (2023) A Survey of Text Representation and Embedding Techniques in NLP. *IEEE Access*, 11, pp.36120–36146.

Peinelt, N., Nguyen, D. and Liakata, M., (2020) *tBERT: Topic Models and BERT Joining Forces for Semantic Similarity Detection*.

Pennington, J., Socher, R. and Manning, C.D., (2014) *GloVe: Global Vectors for Word Representation*. [online] Available at: http://nlp.

Qaiser, S. and Ali, R., (2018) Text Mining: Use of TF-IDF to Examine the Relevance of Words to Documents. *International Journal of Computer Applications*, 1811, pp.25–29.

Rosu, R., Stoica, A.S., Popescu, P.S. and Mihaescu, M.C., (2020) NLP based Deep Learning Approach for Plagiarism Detection. *International Joural of User-System Interaction*, 131, pp.48–60.

Saeed, A.A.M. and Taqa, A.Y., (2022a) A proposed approach for plagiarism detection in Article documents. *SinkrOn*, 72, pp.568–578.

Saeed, A.A.M. and Taqa, A.Y., (2022b) Textual Plagiarism Detection Using Embedding Models and Siamese LSTM. In: *2022 International Conference for Natural and Applied Sciences, ICNAS 2022*. Institute of Electrical and Electronics Engineers Inc., pp.95–100.

Saqaabi, A.A.L., Akrida, E., Cristea, A. and Stewart, C., (2022) A Paraphrase Identification Approach in Paragraph Length Texts. In: *IEEE International Conference on Data Mining Workshops, ICDMW*. IEEE Computer Society, pp.358–367.

Setha, I. and Aliane, H., (2022) Enhancing automatic plagiarism detection: Using Doc2vec model. In: *ICAASE 2022 - 5th Edition of the International Conference on Advanced Aspects of Software Engineering, Proceedings*. Institute of Electrical and Electronics Engineers Inc.

Sharma, L., Graesser, L., Nangia, N. and Evci, U., (2019) *Natural Language Understanding with the Quora Question Pairs Dataset*. [online] Available at: https://www.kaggle.com/c/quora-question-pairs.

Sharma, L., Graesser, L., Nangia, N. and Evci, U., (n.d.) *Natural Language Understanding with the Quora Question Pairs Dataset*.

Thakur, N., Reimers, N., Daxenberger, J. and Gurevych, I., (2021) Augmented SBERT: Data Augmentation Method for Improving Bi-Encoders for Pairwise Sentence Scoring Tasks. *NAACL-HLT 2021 - 2021 Conference of the North American Chapter of the Association for*

Computational Linguistics: Human Language Technologies, Proceedings of the Conference, 2, pp.296–310.

Vasuteja, A., Reddy, A.V. and Pravin, A., (2024) Beyond Copy Paste: Plagiarism Detection using Machine Learning. In: *7th International Conference on Inventive Computation Technologies, ICICT 2024*. Institute of Electrical and Electronics Engineers Inc., pp.245–251.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, L. and Polosukhin, I., (2017a) Attention Is All You Need. [online] Available at: http://arxiv.org/abs/1706.03762.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, L. and Polosukhin, I., (2017b) Attention Is All You Need.

Wang, M., Xie, P., Du, Y. and Hu, X., (2023) T5-Based Model for Abstractive Summarization: A Semi-Supervised Learning Approach with Consistency Loss Functions. *Applied Sciences (Switzerland)*, 1312.

Wang, X., Li, C., Zheng, Z. and Xu, B., (2018) *Paraphrase Recognition via Combination of Neural Classifier and Keywords*. IEEE.

Warner, B., Chaffin, A., Clavié, B., Weller, O., Hallström, O., Taghadouini, S., Gallagher, A., Biswas, R., Ladhak, F., Aarsen, T., Cooper, N., Adams, G., Howard, J. and Poli, I., (2024) Smarter, Better, Faster, Longer: A Modern Bidirectional Encoder for Fast, Memory Efficient, and Long Context Finetuning and Inference. [online] Available at: http://arxiv.org/abs/2412.13663.

Yenduri, G., M, R., G, C.S., Y, S., Srivastava, G., Maddikunta, P.K.R., G, D.R., Jhaveri, R.H., B, P., Wang, W., Vasilakos, A. V. and Gadekallu, T.R., (2023) Generative Pre-trained Transformer: A Comprehensive Review on Enabling Technologies, Potential Applications, Emerging Challenges, and Future Directions. [online] Available at: http://arxiv.org/abs/2305.10435.

Zhang, J., Xue, S., Li, J.L. and She, J., (2022) Automated Plagiarism Detection Model Based On Deep Siamese Network. In: *Proceedings of 2022 8th IEEE International Conference on Cloud Computing and Intelligence Systems, CCIS 2022*. Institute of Electrical and Electronics Engineers Inc., pp.298–302.

Zhang, X., Rong, W., Liu, J., Tian, C. and Xiong, Z., (2017) *Convolution Neural Network Based Syntactic and Semantic Aware Paraphrase Identification*. IEEE.

Zhang, Y., Baldridge, J. and He, L., (2019a) PAWS: Paraphrase Adversaries from Word Scrambling. [online] Available at: http://arxiv.org/abs/1904.01130.

Zhang, Y., Baldridge, J. and He, L., (2019b) PAWS: Paraphrase Adversaries from Word Scrambling.