PARAPHRASE IDENTIFICATION

ALA, VENKATA NITHYA ala.v@northeastern.edu

DONEPUDI, LAKSHMI SOWMYA donepudi.l@northeastern.edu

VURA, SHRAVYA vura.s@northeastern.edu

Abstract

The goal of this project is to develop a reliable paraphrase identification system by using Long Short-Term Memory networks (LSTMs) to identify semantic similarities between text parts. Achieving high accuracy in identifying text pair equivalency requires a detailed examination of syntactic and semantic subtleties. A deep bidirectional LSTM network (DBLSTM) was designed to identify semantic similarity between pairs of sentences and perform binary classification to predict whether the pair of sentences are paraphrases or not. The widely known Microsoft Research Paraphrase Corpus, which is renowned for its different degrees of paraphrasing, has been used for training and evaluation. Interestingly, this effort intends to attain comparable performance without the requirement for high-speed GPUs, whereas the existing state-of-the-art depends on transformer-based structures and pre-trained language models like BERT, thereby providing a more approachable solution that may have consequences for a variety of fields, including text summarization, information retrieval, question-answering, and plagiarism detection.

Introduction

The task of paraphrase detection is very important in the constantly expanding field of natural language processing (NLP). Finding the semantic similarity between two sentences is known as paraphrase detection, and it makes applications like question-answering systems, information retrieval, and duplicate content recognition (plagiarism detection) more efficient. One of the most important components in comprehending and improving the capabilities of language processing and understanding systems is accurate paraphrase identification.

By presenting a neural network architecture with a deep Bidirectional Long Short-Term Memory (BiLSTM) framework, this project aims to perform paraphrase detection between pairs of sentences. Siamese networks were used to compare two input sentences so that the model can determine whether they have similar meanings. Using BiLSTM layers made it easier to extract context from words that come before (preceding) and after (following) one another in a phrase.

The suggested model has been trained and evaluated using data from the Microsoft Research Paraphrase Corpus (MSRP) [1]. The dataset provides a wide range of linguistic variances for the model to handle. It consists of pairs of sentences tagged as paraphrases or non-paraphrases. To prepare input sentences for meaningful representation learning, the project also presents data pre-processing techniques such as tokenization and a defined vocabulary.

The design of Bidirectional Long Short-Term Memory (BiLSTM) layers involved a meticulous exploration of various hyperparameters. Through systematic experimentation, the optimal configuration of

BiLSTM units, layers, and other architectural parameters was determined, aiming to enhance the model's capacity to capture intricate semantic relationships within sentence pairs. The proposed model's effectiveness has been rigorously evaluated through comprehensive training and testing. This evaluation encompasses the analysis of performance metrics, including accuracy, F1-score, and area under Receiver Operating Characteristic (AUC-ROC) curves. Additionally, the exploration of regularization techniques (L2 regularization) has been included in the study. L2 regularization was incorporated with an aim to mitigate overfitting, foster generalization, and promote the learning of robust features. The effects of L2 regularization on the model's performance have been examined and discussed, contributing valuable insights to the broader discourse on optimizing neural network architectures for paraphrase detection in natural language processing tasks.

Background

The current state of the art in paraphrase identification showcases a growing emphasis on transformer-based structures and Attentive Neural Networks [2]. Numerous scholarly studies have introduced sophisticated paraphrase identification systems, the training and evaluation of which have been conducted using the Microsoft Research Paraphrase (MSRP) Corpus. Research conducted by notable scholars in the field of NLP such as *Jacob Eisenstein*, *Yangfeng Ji*,[3] *William B. Dolan, et al*[4] utilized MSRP for developing and testing robust models for tasks not only limited to paraphrase identification but also paraphrase extraction and paraphrase generation. There has been a notable trend towards the integration of pre-trained language models, such as BERT (Bidirectional Encoder Representations from Transformers), for a more nuanced understanding of sentence semantics and context. However, the state-of-the-art solutions rely on high-performance GPUs [5]. By exploring the advantages of these systems, we aim to develop a robust model that does not require a powerful GPU to perform accurate paraphrase identification.

Approach

The model was designed to meticulously analyze pairs of input sentences and discern semantic similarities through an intricate process of embedding, contextual processing, and classification by following the approach below:

Data Pre-processing:

The MSRP corpus data consists of a text file with 5800 sentence pairs that were taken from online news sources and annotated by humans to indicate whether each pair represents a paraphrase or semantic equivalency relationship.

1) Data Setup:

- The MSRP training and testing data was loaded.

2) Tokenizing Sentences:

- Sentences are tokenized into words using a custom function. This function handles things like lowercase formatting and removing punctuation marks.

3) Building Vocabulary:

- A vocabulary is created by going through all the sentences.
- For each word, its frequency in the corpus is calculated.

4) Mapping Word IDs:

- Each word is mapped to a unique numerical identifier.
- Special tokens such as ‘sentence_start’ and ‘sentence_end’ are also mapped to specific numerical IDs.
- The vocabulary size can be controlled by setting a limit on the number of considered words.

5) Preparing Data for Model:

- All sentences for training and testing are converted into sequences of these IDs.
- Words not in the vocabulary are replaced with a special "unknown" token.
- Sequences are adjusted to have the same length for easier processing by the model.

BiLSTM Model:

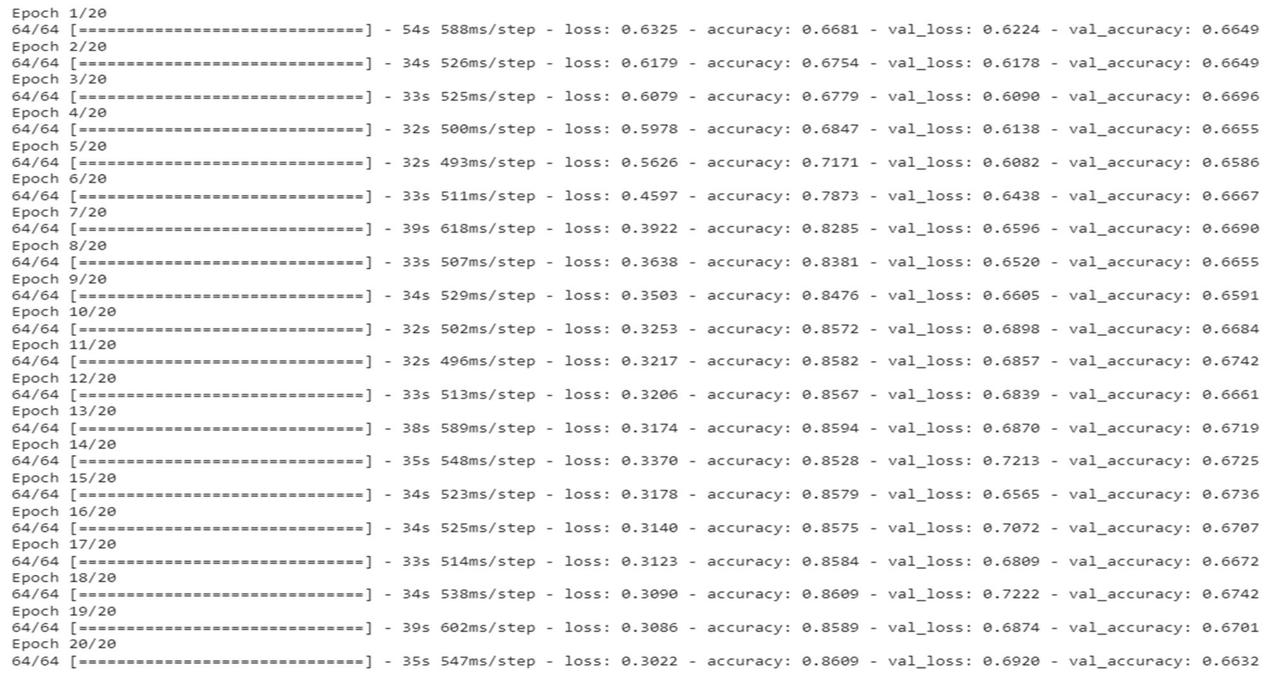
Inputs: The input layer of the model comprises two distinct sentences, denoted as input_sentence1 and input_sentence2. These sentences undergo embedding via a shared layer, converting words into dense vectors. The embedding process results in sequences of vectors, creating embedded_sentence1 and embedded_sentence2, which serve as the foundation for subsequent analysis.

Bidirectional LSTM Layers: Two bidirectional LSTM layers have been employed. The first layer, lstm_layer1, a Bidirectional LSTM with 100 units, is configured to return sequences, enabling the generation of context vectors for each word in the input sentences. The subsequent layer, lstm_layer2, a Bidirectional LSTM with 50 units, does not return sequences but instead produces a single vector representing the context of the entire sentence.

Context Vectors: The output of the Bidirectional LSTM layers yields context vectors for each word in the input sentences. These vectors are encapsulated in output_sentence1 and output_sentence2, capturing the nuanced contextual information essential for discerning paraphrastic relationships.

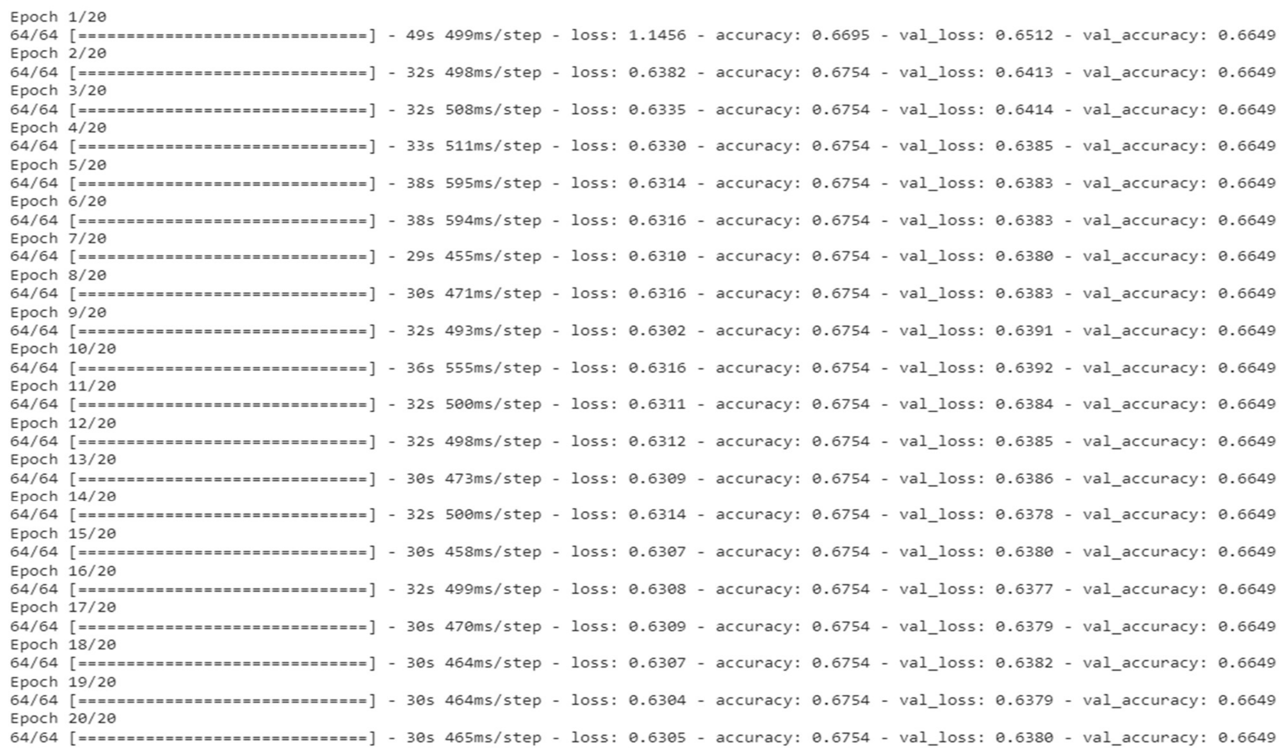
Concatenation and Dense Layers: The context vectors are systematically concatenated to create a merged vector, denoted as merged_output. This merged representation undergoes further analysis through dense layers. The initial dense layer (dense_layer) comprises 64 units with Rectified Linear Unit (ReLU) activation, facilitating the extraction of intricate features from the concatenated context vectors. The subsequent dense layer (output_layer) employs Sigmoid activation, yielding a binary classification indicating whether the input sentences are paraphrases or not.

Regularization: To explore the effects of L2 regularization, it has been strategically incorporated at various layers to mitigate potential overfitting and enhance the generalization capabilities. The BiLSTM layers, the intermediate dense layer (dense_layer) and the final output layer (output_layer) have been subject to L2 regularization. The regularization strength, denoted by the parameter l2_lambda, was introduced to impose a penalty on the model's weights during training. Different strengths of the hyperparameter l2_lamda have been incorporated, and it was observed that for l2_lamda = 0.01, the model performance was best. However, despite the inclusion of regularization techniques, empirical findings from model training on the dataset indicate that, in this specific context, regularization is not effective in improving the model's performance. Because the dataset is clean and well-labeled, and there is little noise in the training data, regularization does not provide substantial benefits.



```
Epoch 1/20
64/64 [==============================] - 54s 588ms/step - loss: 0.6325 - accuracy: 0.6681 - val_loss: 0.6224 - val_accuracy: 0.6649
Epoch 2/20
64/64 [==============================] - 34s 526ms/step - loss: 0.6179 - accuracy: 0.6754 - val_loss: 0.6178 - val_accuracy: 0.6649
Epoch 3/20
64/64 [==============================] - 33s 525ms/step - loss: 0.6079 - accuracy: 0.6779 - val_loss: 0.6090 - val_accuracy: 0.6696
Epoch 4/20
64/64 [==============================] - 32s 500ms/step - loss: 0.5978 - accuracy: 0.6847 - val_loss: 0.6138 - val_accuracy: 0.6655
Epoch 5/20
64/64 [==============================] - 32s 493ms/step - loss: 0.5626 - accuracy: 0.7171 - val_loss: 0.6082 - val_accuracy: 0.6586
Epoch 6/20
64/64 [==============================] - 33s 511ms/step - loss: 0.4597 - accuracy: 0.7873 - val_loss: 0.6438 - val_accuracy: 0.6667
Epoch 7/20
64/64 [==============================] - 39s 618ms/step - loss: 0.3922 - accuracy: 0.8285 - val_loss: 0.6596 - val_accuracy: 0.6690
Epoch 8/20
64/64 [==============================] - 33s 507ms/step - loss: 0.3638 - accuracy: 0.8381 - val_loss: 0.6520 - val_accuracy: 0.6655
Epoch 9/20
64/64 [==============================] - 34s 529ms/step - loss: 0.3503 - accuracy: 0.8476 - val_loss: 0.6605 - val_accuracy: 0.6591
Epoch 10/20
64/64 [==============================] - 32s 502ms/step - loss: 0.3253 - accuracy: 0.8572 - val_loss: 0.6898 - val_accuracy: 0.6684
Epoch 11/20
64/64 [==============================] - 32s 496ms/step - loss: 0.3217 - accuracy: 0.8582 - val_loss: 0.6857 - val_accuracy: 0.6742
Epoch 12/20
64/64 [==============================] - 33s 513ms/step - loss: 0.3206 - accuracy: 0.8567 - val_loss: 0.6839 - val_accuracy: 0.6661
Epoch 13/20
64/64 [==============================] - 38s 589ms/step - loss: 0.3174 - accuracy: 0.8594 - val_loss: 0.6870 - val_accuracy: 0.6719
Epoch 14/20
64/64 [==============================] - 35s 548ms/step - loss: 0.3370 - accuracy: 0.8528 - val_loss: 0.7213 - val_accuracy: 0.6725
Epoch 15/20
64/64 [==============================] - 34s 523ms/step - loss: 0.3178 - accuracy: 0.8579 - val_loss: 0.6565 - val_accuracy: 0.6736
Epoch 16/20
64/64 [==============================] - 34s 525ms/step - loss: 0.3140 - accuracy: 0.8575 - val_loss: 0.7072 - val_accuracy: 0.6707
Epoch 17/20
64/64 [==============================] - 33s 514ms/step - loss: 0.3123 - accuracy: 0.8584 - val_loss: 0.6809 - val_accuracy: 0.6672
Epoch 18/20
64/64 [==============================] - 34s 538ms/step - loss: 0.3090 - accuracy: 0.8609 - val_loss: 0.7222 - val_accuracy: 0.6742
Epoch 19/20
64/64 [==============================] - 39s 602ms/step - loss: 0.3086 - accuracy: 0.8589 - val_loss: 0.6874 - val_accuracy: 0.6701
Epoch 20/20
64/64 [==============================] - 35s 547ms/step - loss: 0.3022 - accuracy: 0.8609 - val_loss: 0.6920 - val_accuracy: 0.6632
```

Fig.1 Model history before regularization (better performance)



```
Epoch 1/20
64/64 [==============================] - 49s 499ms/step - loss: 1.1456 - accuracy: 0.6695 - val_loss: 0.6512 - val_accuracy: 0.6649
Epoch 2/20
64/64 [==============================] - 32s 498ms/step - loss: 0.6382 - accuracy: 0.6754 - val_loss: 0.6413 - val_accuracy: 0.6649
Epoch 3/20
64/64 [==============================] - 32s 508ms/step - loss: 0.6335 - accuracy: 0.6754 - val_loss: 0.6414 - val_accuracy: 0.6649
Epoch 4/20
64/64 [==============================] - 33s 511ms/step - loss: 0.6330 - accuracy: 0.6754 - val_loss: 0.6385 - val_accuracy: 0.6649
Epoch 5/20
64/64 [==============================] - 38s 595ms/step - loss: 0.6314 - accuracy: 0.6754 - val_loss: 0.6383 - val_accuracy: 0.6649
Epoch 6/20
64/64 [==============================] - 38s 594ms/step - loss: 0.6316 - accuracy: 0.6754 - val_loss: 0.6383 - val_accuracy: 0.6649
Epoch 7/20
64/64 [==============================] - 29s 455ms/step - loss: 0.6310 - accuracy: 0.6754 - val_loss: 0.6380 - val_accuracy: 0.6649
Epoch 8/20
64/64 [==============================] - 30s 471ms/step - loss: 0.6316 - accuracy: 0.6754 - val_loss: 0.6383 - val_accuracy: 0.6649
Epoch 9/20
64/64 [==============================] - 32s 493ms/step - loss: 0.6302 - accuracy: 0.6754 - val_loss: 0.6391 - val_accuracy: 0.6649
Epoch 10/20
64/64 [==============================] - 36s 555ms/step - loss: 0.6316 - accuracy: 0.6754 - val_loss: 0.6392 - val_accuracy: 0.6649
Epoch 11/20
64/64 [==============================] - 32s 500ms/step - loss: 0.6311 - accuracy: 0.6754 - val_loss: 0.6384 - val_accuracy: 0.6649
Epoch 12/20
64/64 [==============================] - 32s 498ms/step - loss: 0.6312 - accuracy: 0.6754 - val_loss: 0.6385 - val_accuracy: 0.6649
Epoch 13/20
64/64 [==============================] - 30s 473ms/step - loss: 0.6309 - accuracy: 0.6754 - val_loss: 0.6386 - val_accuracy: 0.6649
Epoch 14/20
64/64 [==============================] - 32s 500ms/step - loss: 0.6314 - accuracy: 0.6754 - val_loss: 0.6378 - val_accuracy: 0.6649
Epoch 15/20
64/64 [==============================] - 30s 458ms/step - loss: 0.6307 - accuracy: 0.6754 - val_loss: 0.6380 - val_accuracy: 0.6649
Epoch 16/20
64/64 [==============================] - 32s 499ms/step - loss: 0.6308 - accuracy: 0.6754 - val_loss: 0.6377 - val_accuracy: 0.6649
Epoch 17/20
64/64 [==============================] - 30s 470ms/step - loss: 0.6309 - accuracy: 0.6754 - val_loss: 0.6379 - val_accuracy: 0.6649
Epoch 18/20
64/64 [==============================] - 30s 464ms/step - loss: 0.6307 - accuracy: 0.6754 - val_loss: 0.6382 - val_accuracy: 0.6649
Epoch 19/20
64/64 [==============================] - 30s 464ms/step - loss: 0.6304 - accuracy: 0.6754 - val_loss: 0.6379 - val_accuracy: 0.6649
Epoch 20/20
64/64 [==============================] - 30s 465ms/step - loss: 0.6305 - accuracy: 0.6754 - val_loss: 0.6380 - val_accuracy: 0.6649
```

Fig.2 Model history after L2 regularization (no improvement)

Model Output: The final output is a binary classification, providing a conclusive indication of the paraphrastic relationship between the input sentences. This architecture integrates advanced neural network components to ensure the model's proficiency in capturing semantic nuances, contributing to efficient paraphrase detection in natural language processing.

Results

The experimental evaluation was conducted on the Microsoft Research Paraphrase Corpus (MSRP)[1] dataset, a widely recognized benchmark for paraphrase detection tasks. The dataset consists of sentence pairs labeled as paraphrases (class 1) or non-paraphrases (class 0), providing a diverse set of linguistic variations for rigorous evaluation.

Experimental Setup: The BiLSTM model was trained and tested using a carefully curated set of preprocessed sentences derived from the MSRP dataset. Tokenization, vocabulary mapping, and sequence padding were performed as detailed in the data preprocessing section, ensuring consistent and meaningful representation of textual information.

Performance Metrics: The classification report presents a comprehensive overview of the model's performance. The precision, recall, and F1-score metrics are reported for both paraphrase (class 1) and non-paraphrase (class 0) categories. The model exhibits notable precision (0.67) and recall (0.97) for paraphrases, indicating a strong ability to correctly identify instances of semantic similarity. However, the precision for non-paraphrases is relatively lower (0.47), suggesting a challenge in distinguishing non-paraphrastic relationships. The overall accuracy stands at 0.66, with an area under the ROC curve of 0.67.

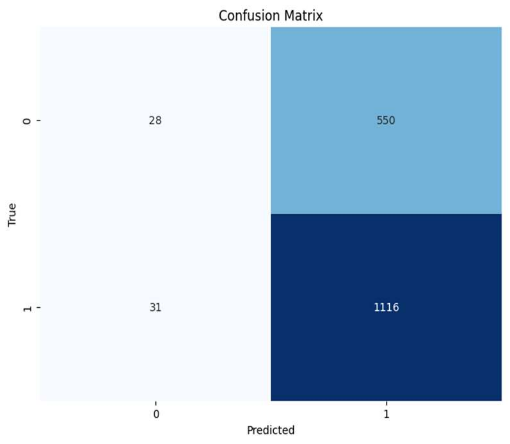


Fig.3 Confusion Matrix

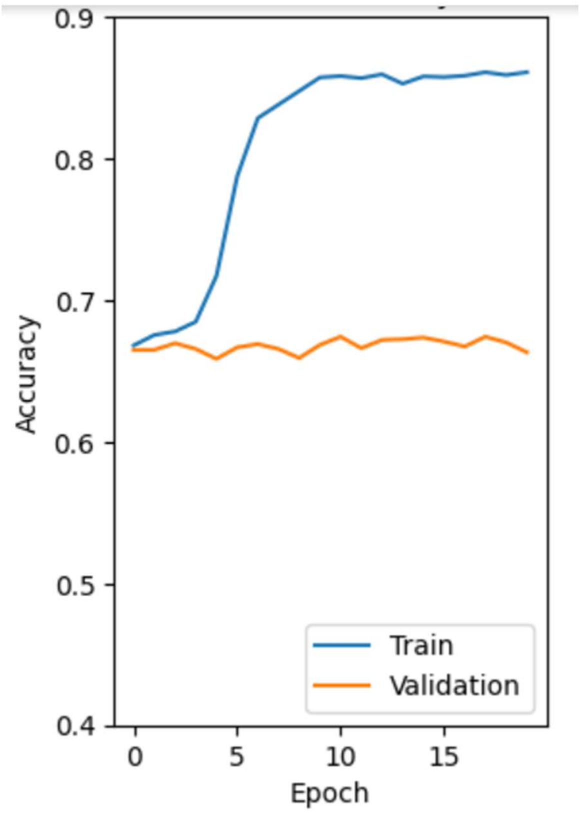
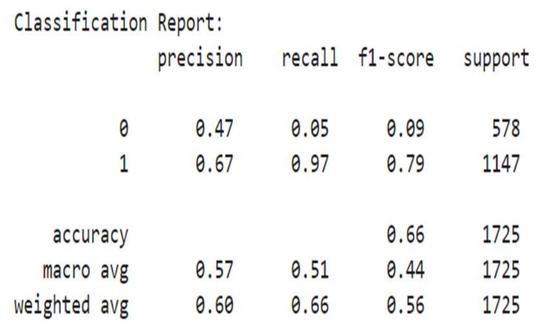


Fig.4 Training and Validation Accuracy plot



```
Classification Report:
              precision    recall  f1-score   support

           0       0.47      0.05      0.09       578
           1       0.67      0.97      0.79      1147

    accuracy                           0.66      1725
   macro avg       0.57      0.51      0.44      1725
weighted avg       0.60      0.66      0.56      1725
```

Fig.5 Classification Report

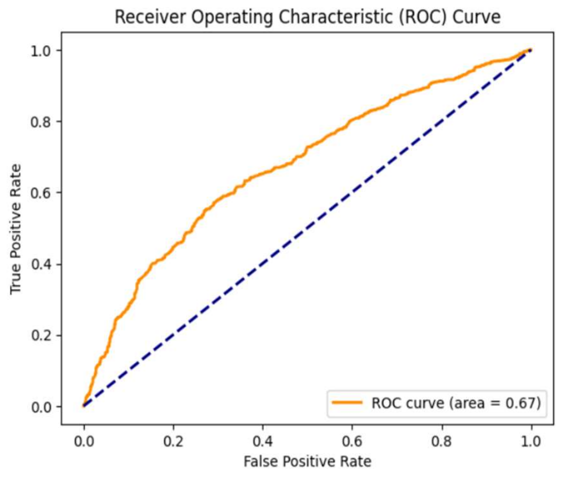


Fig.6 ROC curve

Discussion: The results highlight the model's effectiveness in detecting paraphrases, particularly with a high recall rate, ensuring a comprehensive identification of true positives. However, the lower precision for non-paraphrases indicates potential areas for improvement, possibly through further addition of data. The macro-average F1-score (0.44) suggests a balance between precision and recall and the area under ROC curve (0.67) suggests that the model is a fairly good paraphrase identifier.

In summary, the presented results demonstrate the model's capability to discern paraphrastic relationships with promising accuracy. To enhance the model's overall performance, future directions may involve fine-tuning hyperparameters, exploring alternative architectures, or incorporating external linguistic features. Additionally, addressing the nuances of non-paraphrastic relationships to improve precision in this category represents an avenue for further investigation.

In a broader context, this study contributes to the ongoing discourse in paraphrase detection, emphasizing the practical implications of achieving a nuanced understanding of semantic relationships within sentences. The presented results and insights contribute to the broader field of natural language processing, fostering advancements in the development of models that do not demand high computational capacities.

Conclusion

In this paper, a computationally efficient paraphrase identification method that leverages a complex neural network architecture enriched with 2 bidirectional long- and short-term memory (BiLSTM) layers is presented. The model was meticulously designed and trained on the Microsoft Research Paraphrase Corpus (MSRP) dataset, a gold standard in paraphrase detection tasks. The comprehensive evaluation highlighted the performance of the model in recognizing clear semantic similarities, especially with high expression recall rates and a fairly good accuracy. Incorporating L2 regularization has been explored to improve generalization, although empirical results suggest that it is not necessary for this particular corpus.

The results provide valuable insights into the field of natural language processing, highlighting the complex challenges of semantic interpretation and the potential effectiveness of BiLSTM networks.

The resulting performance metrics such as accuracy, recall and area under ROC curve demonstrate the practical utility of the model in real-world applications that require a deep understanding of semantic relationships in sentences. While the presented model shows promising capabilities, other research directions include fine-tuning hyperparameters, exploring alternative regularization strategies, and investigating additional data with rich linguistic features, specifically for distinguishing non-paraphrastic relationships. This study presents a solution for paraphrase identification that is at par with models proposed on high performance GPUs. The key challenge of identifying paraphrases without the need of high computational capacities is successfully addressed.

References

[1] Microsoft Research Paraphrase Corpus, *Microsoft,* https://www.microsoft.com/en-us/download/details.aspx?id=52398

[2] Modelling Sentence Pairs with Tree-structured Attentive Encoder, *Yao Zhou, Cong Liu, Yan Pan,* https://arxiv.org/pdf/1610.02806.pdf

[3] Discriminative Improvements to Distributional Sentence Similarity, *Yangfeng Ji, Jacob Eisenstein,* https://aclanthology.org/D13-1090.pdf

[4] Automatically Constructing a Corpus of Sentential Paraphrases, *William B. Dolan and Chris Brockett,* https://aclanthology.org/I05-5002.pdf

[5] Paraphrase Identification with Lexical, Syntactic and Sentential Encodings, *Sheng Xu, Xingfa Shen, Fumiyo Fukumoto, Jiyi Li Yoshimi Suzuki, and Hiromitsu Nishizaki,* https://www.mdpi.com/2076-3417/10/12/4144