A Deep Learning Based Approach to Paraphrase Detection

[†]Keerthana Kodaiarasu, and [‡]Likhith Muthavarappu
[†]Department of Computer Science, California State University, Sacramento, CA 95819, USA
Email: keerthanakodaiarasu@csus.edu, likhithmuthavarapu@csus.edu

***Abstract*—Rapid advancements in Artificial Intelligence(AI) have brought about great developments in the fields of Natural Language Processing (NLP) and text generation. While the progress in these fields have received overwhelming response it also raises a concern regarding the credibility of the original idea in the paraphrased text. Though there has been significant progress towards developing paraphrase identification mechanisms and tools that can identify paraphrased content to an extent they still are unable to identify most complex sentence structures. Hence, they lack the ability to distinguish original and paraphrased text. In this paper, we propose a deep learning based approach to paraphrase identification and detection. We are using the Microsoft Research Paraphrase Corpus (MRPC) dataset for model implementation and evaluation. We start by preprocessing the data by performing stemming and lemmatization. Following this steps we create vectors of three different similarity scores for each pair of sentences. Using the vectors of similarity scores from string, semantic and embedding similarities we train and evaluate different deep learning models.**

I. INTRODUCTION

NLP and text generation have greatly advanced as a result of the quick growth of AI and Machine Learning (ML). With the development of language models, there has been a massive improvement in understanding and even generate human-like text. Some examples of such systems are voice assistants, speech recognition, machine translation, text-to-speech translation, text generation etc. Language models are trained on massive amounts of text data, such as books, articles, and social media posts, to learn the statistical patterns of language and build a probabilistic model of how words and phrases relate to each other. They can learn to recognise subtle linguistic patterns and context-dependant patterns. Similarly, paraphrase detection is the task of identifying if two different pieces of text convey the same idea even though they may be worded differently. This problem is of significant interest in artificial intelligence and has many practical applications such as plagiarism detection, text summarizing, machine translation and question answering systems.

Though there are has been significant development in this area, paraphrase detection is the most challenging task because it requires an understanding the semantics of the text rather than just matching the sentences or phrases word by word. There are a lot of researches going on this particular field and researchers have developed various approaches to tackle this problem, including rule-based methods, statistical methods, machine learning algorithms. The development of effective paraphrase detection techniques is critical for advancing the field of natural language processing and improving the accuracy of many downstream applications.

In this paper, we apply different deep learning based approaches to solve the above cited problem. In this project, we utilize pair of sentences, that is the original text and the paraphrased text for training and validation. To measure the similarity between two sentences or phrases generally there are three kinds of similarity techniques used. The first technique is String Similarity which works by comparing the two sentences as a whole or character by character and scoring them. The second technique is called Semantic Similarity which works by measuring the scores of similarity by comparing the meaning of the sentences. The final technique is called Embedding Similarity technique often calculate the similarity score by converting the two sentences into two vectors, comparing those two vectors, and then computing the result.

This paper is structured as shown below:
1. We will first introduce the dataset and different pre-processing techniques we have been leveraging on the dataset.
2. Second, we will review the different similarity techniques and discuss the implementation strategy
3. Finally, we will evaluate the performance of our deep learning models by passing the vectors obtained in the previous section.

We conclude this research by summarizing the main contributions of the paper for implementing an deep learning based approaches and presenting comparative analysis with our base reference model.

II. RELATED WORK

In paper [2] the author addresses the challenge of measuring how similar two sentences are. The authors propose a method called Siamese Recurrent Architecture (SRA) that uses two identical Recurrent Neural Networks (RNNs) to represent the sentences as fixed-length vectors. These vectors are then fed into a similarity function to output a score between 0 and 1, indicating the degree of similarity. The authors show that their model outperforms other methods on two datasets and can be used for tasks such as paraphrase detection and textual entailment.

BERT[3] is a brand-new language representation model that takes into account both left and right context to pre-train deep bidirectional representations. For many NLP applications, it can be improved with just one extra output layer to produce cutting-edge models. On NLP tasks, BERT has achieved cutting-edge results.

An unsupervised approach for learning to generate paraphrases using multiple-sequence alignment [4]. The proposed approach models the relationship between original sentences and their paraphrases using a probabilistic framework and is trained on a large corpus of aligned sentence pairs. Experimental results show that the proposed approach can generate high-quality paraphrases that are comparable in quality to those generated by supervised approaches, and that it outperforms existing unsupervised approaches.

A similar paper introduces a model called Recursive Neural Tensor Network (RNTN) for sentiment analysis that can understand the meaning of phrases and sentences[5]. The model is built using a recursive neural network and a tensor-based composition function to capture interactions between multiple word vectors. The authors evaluate the model on the Stanford Sentiment Treebank dataset and show that it outperforms several baseline models, achieving state-of-the-art results. The paper also provides insights into the contribution of different model components, highlighting the importance of the tensor-based composition function and the recursive structure of the model.

The base reference model for our suggested approach is from the paper paper that proposes a model that combines three similarity techniques (string, semantic, and embedding) and feeds the resulting scores to machine learning classifiers[1].

III. PROBLEM FORMULATION

In this paper, we aim to solve one of the most important problems in language modelling which is the paraphrase detection problem. In this project, we are using the Microsoft Research Paraphrase Corpus (MRPC) dataset as input which consists of 5801 pairs of sentences (original sentence and paraphrased sentence) with each pair being valued as semantically equivalent or not. The dataset consists of 4076 sentence pairs in training set and 1725 pairs of sentences in testing set. Accompanying each pair of sentence is a judgement indicating whether several human annotators thought the two sentences were similar enough to be deemed as close phrases. We preprocess the dataset by applying stemming techniques to stem or remove unnecessary characters from the words and lemmatize to understand the underlying context in the sentence and convert them to a semantically equivalent base form. After preprocessing the dataset we use various similarity scoring techniques to generate feature scores. After generating the feature scores separately, we stack them and apply deep learning models such as Fully Connected Neural Networks (FCNN), Convolutional Neural Networks (CNN) etc., for model training and

Sentence Pairs	Number of Records
Total	5801
Training Set	4076
Testing Set	1725

Fig. 1. Microsoft Research Paraphrase Corpus (MRPC) dataset

evaluation. Finally, we compare and analyze the results with our base model of reference.

IV. SYSTEM DESIGN

A. SYSTEM ARCHITECTURE

In this section, we describe our deep learning model architecture in depth. Figure 2 depicts the system architecture. We take the input, which in our case is the MRPC dataset containing 5801 pairs of sentences with each pair accompanied by a binary judgement that indicates whether that pair is semantically equivalent or not. We then pass the data to a data preprocessing phase where the data is Stemmed and Lemmatized to correct and convert the words in the sentences to understand the meaning of the underlying sentence structure. After this stage, we create feature vectors using the similarity scores from three different Similarity techniques. We use String similarity to score sentence pairs based on the common sequence of characters or phrases in a sentence, Semantic similarity technique scores sentences based on the semantic equivalence between each pair of sentences and embedding similarity technique to convert the two sentences in to vectors and compare them. After converting them to feature vectors, we stack the features together and pass them to deep learning models for training and evaluation.

B. MODEL IMPLEMENTATION

The main aim of the project is to build a robust deep learning model that is capable of classifying whether a pair of sentence is paraphrased or not. One approach to solve this problem is to create a model that takes in two texts as input and outputs a binary label indicating whether they are paraphrases or not. We started by collecting the dataset of our reference paper which is the Microsoft Research Paraphrase Corpus (MRPC). The dataset is divided into two parts which is the train and test dataset. Both datasets consists of two sentences and a label which classifies whether each pair of sentence is a paraphrased text or dissimilar text.

The system architecture is divided into three phases. Stage 1 is the data preprocessing stage. Stage 2 is to Create and stack the similarity scoring functions. The Final and last stage, Stage 3 is to use these similarity scores to train the models.

Step 1 - Data Preprocessing: In this step, the raw text data is preprocessed to remove noise and standardize the input format. The preprocessing steps include removing punctuations, converting to lowercase, tokenizing, removing stop words, removing numbers and special characters. This step helps in reducing the noise in the text data and making it easier for the model to learn. For the purpose of paraphrase detection we incorporate both stemming and lemmatization step in text preprocessing step. Stemming can be used to reduce the number of unique words in the text data, which can make it easier to process and reduce the dimensionality of the feature space. Lemmatization can be used to produce meaningful lemmas for the words, which can help improve the accuracy of some models. For example, if the input text data includes words like ”running”, ”runs”, and ”ran”, stemming could reduce all three to ”run”. Similarly, lemmatization could reduce them to the lemma ”run” as well. These techniques can help reduce the impact of the different word forms on the performance of the models, which can improve their accuracy.

Step 2 - Creating a stack of similarity scoring functions: In this step, a stack of similarity scoring functions is created to

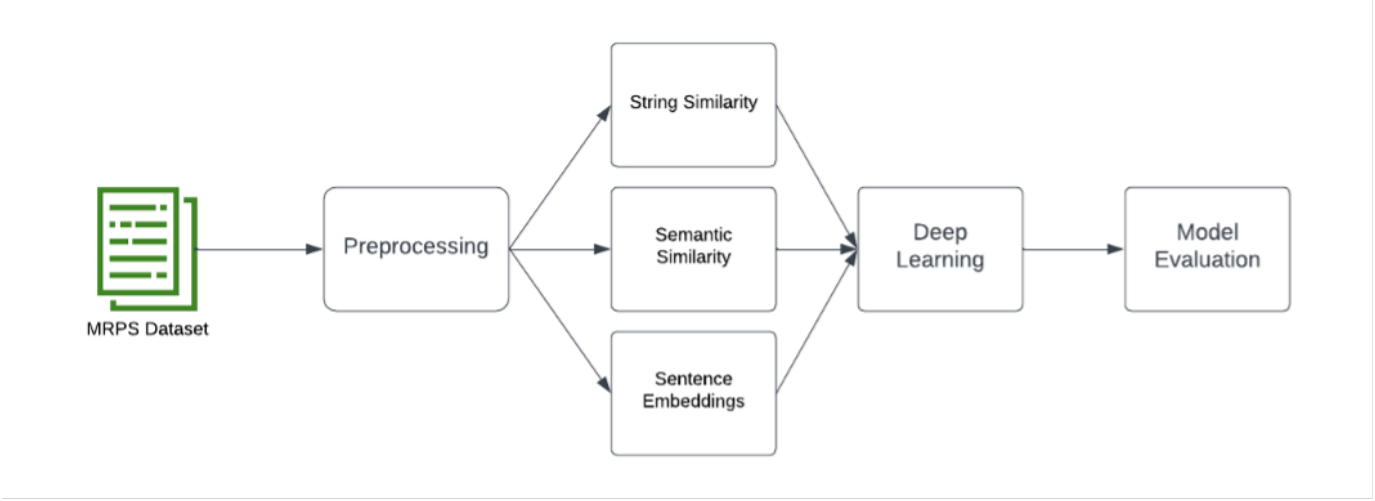

Fig. 2. System Architecture

compute different kinds of similarity scores between the two texts. The similarity scores are then used as features to train the model.

The three similarity scoring functions used in this project are:
1. **String similarity score:** This score is 1 if the two texts are identical, and 0 otherwise. For implementing String Similarity scoring techniques we used **textdistance** which is a python library for computing distance metrics between two strings. It provides a variety of distance metrics that can be used to compare two strings based on their similarity or dissimilarity. Some of the distance metrics provided by text distance include but not limited to Hamming distance, Levenshtein distance, Jaro-Winkler distance etc. We used a total of 27 textdistance metrics for generating similarity scores for each pair of sentences.

2. **Semantic similarity score:** This score is computed using pre-trained word embeddings to measure the semantic similarity between the two texts. We used **GloVE** embeddings for measuring the scoring of semantic similarity. GloVE embeddings is a popular unsupervised technique for generating word embeddings, which are vector representations of words in a high-dimensional space. GloVe embeddings have several advantages over other types of word embeddings. First, they capture both syntactic and semantic relationships between words, which can improve the performance of NLP models. Second, they can be pre-trained on large corpora of text data, making them suitable for transfer learning across different NLP tasks. Third, they are computationally efficient to compute and require less memory than other types of embeddings, such as contextualized embeddings. We also used another embedding for scoring the semantic similarity measure which is the **SpaCy** embeddings. SpaCy uses a neural network architecture to generate word embeddings. The architecture is based on a convolutional neural network (CNN) and a max-pooling layer, which is trained on a large corpus of text data. SpaCy embeddings have several advantages over other types of word embeddings. First, they capture both syntactic and semantic relationships between words, which can improve the performance of NLP models. Second, they are trained on a large corpus of text data, making them suitable for transfer learning across different NLP tasks. Third, they can be fine-tuned on a smaller corpus of text data to improve their performance on specific tasks.

3. **Embeddings similarity score:** This score is computed using the cosine similarity between the average word embeddings of the two texts. We used ten of the BERT's pretrained models for obtaining similarity scores. We also used Sister embedding for generating similarity scores. Sister (Sentence Representation with Tree-based

```
Model: "model_5"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_6 (InputLayer)        [(None, 40)]              0

 dense_21 (Dense)            (None, 256)               10496

 dropout_11 (Dropout)        (None, 256)               0

 dense_22 (Dense)            (None, 128)               32896

 dropout_12 (Dropout)        (None, 128)               0

 dense_23 (Dense)            (None, 64)                8256

 dropout_13 (Dropout)        (None, 64)                0

 dense_24 (Dense)            (None, 1)                 65

=================================================================
Total params: 51,713
Trainable params: 51,713
Non-trainable params: 0
_________________________________________________________________
```

Fig. 3. Fully Connected Neural Network Model Summary

```
Model: "sequential_2"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv1d_1 (Conv1D)           (None, 40, 64)            128

 max_pooling1d_1 (MaxPooling  (None, 20, 64)           0
 1D)

 flatten_1 (Flatten)         (None, 1280)              0

 dense_16 (Dense)            (None, 128)               163968

 dropout_10 (Dropout)        (None, 128)               0

 dense_17 (Dense)            (None, 1)                 129

=================================================================
Total params: 164,225
Trainable params: 164,225
Non-trainable params: 0
_________________________________________________________________
```

Fig. 4. Convolutional Neural Network Model Summary

```
Model: "model_6"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_7 (InputLayer)        [(None, 40, 1)]           0

 lstm_1 (LSTM)               (None, 128)               66560

 dense_25 (Dense)            (None, 128)               16512

 dense_26 (Dense)            (None, 64)                8256

 dense_27 (Dense)            (None, 1)                 65

=================================================================
Total params: 91,393
Trainable params: 91,393
Non-trainable params: 0
_________________________________________________________________
```

Fig. 5. Long Short Term Memory Model Summary

Inference over Dependency Structures) is a Python library for generating sentence embeddings, which are vector representations of sentences in a high-dimensional space. Sister embeddings are typically used as features for various natural language processing (NLP) tasks, such as text classification, sentiment analysis, and language translation. The algorithm is based on a tree-LSTM network, which is trained on a large corpus of text data.

Step 3 - Training the model on the Similarity Scores: Once we have all the similarity scores from the previous step. In this step, a neural network model is trained on the similarity scores computed in the previous step. Two types of neural network models are used in this project: a convolutional neural network (CNN) and a long short-term memory (LSTM) network. The input to the model is the concatenation of the three similarity scores, and the output is a binary label indicating whether the two texts are paraphrases or not.

Figures 3, 4 and 5 show the model summary for the Deep learning models that we have implemented. We will discussing the performance of each of the models in Section 5 under Evaluation and Results.

V. EVALUATION AND RESULTS

A. METHODOLOGY

In this project, we used the Microsoft Reseach Paraphrase Corpus(MRPC) dataset that consists of 5801 pairs of sentences accompanied by a binary value that indicates whether the sentences are paraphrased or not. The dataset by default consists of training and testing dataset separated. The training data consists of 4076 pairs of sentences whereas the testing dataset consists of 1725 pairs of sentences. This split was made on the original collection with 70 percent for training and 30 percent for testing. The performance of our model was evaluated by generating the classification reports and AU-ROC curve. The classification report consisted of F1-score and Accuracy for model evaluation.

B. RESULTS

In this section, we are presenting the results of our model evaluation. We tried different combinations

Combination	Evaluation Criteria	Classifiers(Logistic, Extra trees classifier, MLP)	FNN	CNN	LSTM
String Similarity (using 27 textdistance algo)	F1-Score		0.8	0.82	0.82
	Accuracy	(0.7341,0.7181,0.7353)	0.71	0.74	0.72
Semantic Similarity (GloVe Embeddings)	F1-Score		0.83	0.83	0.83
	Accuracy	(0.70,0.59,0.70)	0.71	0.71	0.71
Semantic Similarity (SpaCy)	F1-Score		0.83	0.82	0.83
	Accuracy	(0.72,0.63,0.71)	0.71	0.72	0.72
Embedding Similarity (fasttext embeddings)	F1-Score		0.82	0.83	0.82
	Accuracy	(0.74,0.66,0.74)	0.74	0.75	0.74
Embedding Similarity (BERT embeddings)	F1-Score		0.84	0.85	0.85
	Accuracy	(0.78,0.77,0.76)	0.77	0.78	0.77
Combination of all the string and semantic embeddings	F1-Score		0.74	0.75	0.72
	Accuracy	(0.73,0.7328,0.7304)	0.83	0.84	0.8
Combination of all the embeddings	F1-Score		0.76	0.76	0.76
	Accuracy	(0.77,0.76,0.75)	0.84	0.84	0.84

Fig. 6. EVALUATION RESULTS

of similarity scores achieved during the second step of our model implementation. Overall we evaluated combination of 10 BERT embeddings and combination of all similarity scoring techniques gave us good performance compared to base research paper. The evaluation results are presented in Figure 6.

VI. CONCLUSION AND FUTURE WORK

In this paper, we propose a deep learning based approach for paraphrase detection using the Microsoft Research Paraphrase Corpus (MRPC) dataset. We preprocessed the dataset and found the similarity scores for each of the different similarity techniques. After stacking the similarity scores together from different techniques, we trained and evaluated the deep learning models. For Convolutional Neural Network model with 10 BERT embeddings we achieved an overall F1 score of 85 and an accuracy of 78. From the observed results we are able to say that deep learning models have significant improvement over the machine learning classifiers while detecting paraphrased text.

Though the proposed model is successful it is limited to the dataset it is trained on. We aim to improve our current implementation in future by adding some algorithm that can use the similarity scores to identify dissimilar text, partially paraphrased text and completely paraphrased text. Also, we would like to train the dataset on more combinations of similarity techniques for each of String similarity, semantic similarity and embedding similarity.

REFERENCES

[1] Hany, Mena, and Wael H. Gomaa. "A Hybrid Approach to Paraphrase Detection Based on Text Similarities and Machine Learning Classifiers." 2022 2nd International Mobile, Intelligent, and Ubiquitous Computing Conference (MIUCC). IEEE, 2022.

[2] Mueller, J., Thyagarajan, A. (2016). Siamese Recurrent Architectures for Learning Sentence Similarity. Proceedings of the AAAI Conference on Artificial Intelligence, 30(1). https://doi.org/10.1609/aaai.v30i1.10350.

[3] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers), pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

[4] Barzilay, R. and Lee, L. (2003) Learning to Paraphrase: An Unsupervised Approach Using Multiple-Sequence Alignment. Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology, 1, 16-23. https://doi.org/10.3115/1073445.1073448.

[5] Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Ng, and Christopher Potts. 2013. Recursive Deep Models for Semantic Compositionality Over a Sentiment Treebank. In Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, pages 1631–1642, Seattle, Washington, USA. Association for Computational Linguistics.

[6] O'Shea, Keiron, and Ryan Nash. "An introduction to convolutional neural networks." arXiv preprint arXiv:1511.08458 (2015).