EasyChair Preprint

№ 5833

# Comparative Analysis of Semantic Similarity Word Embedding Techniques for Paraphrase Detection

Shrutika Chawla, Preeti Aggarwal and Ravreet Kaur



June 16, 2021

# Comparative analysis of semantic similarity word embedding techniques for paraphrase detection

Shrutika Chawla[1], Preeti Aggarwal[1], Ravreet Kaur[1]

1. UIET, CSE Department, Panjab University, Chandigarh
shrutikachawla.96@gmail.com

**Abstract**

Most of the state of art plagiarism detection tools focus on verbatim reproduction of a document for plagiarism identification (PI) not keeping into account its semantic properties. Recently, deep learning models have shown considerable performance in identifying paraphrases using word-embedding approach. This paper gives an overview and comparison of the performances of five word embedding based deep learning models in the field of semantic similarity detection, such as TF-IDF, Word2Vec, Doc2Vec, FastText and BERT on two publicly available corpora: Quora Question Pairs (QQP) and Plagiarized Short Answers (PSA). After extensive literature review and experiments, the most appropriate text pre-processing approaches, distance measures, and the thresholds have been settled on for detecting semantic similarity/paraphrasing. The paper concludes on FastText being the most efficient model out of the five, both in terms of evaluation metrics i.e., accuracy, precision, recall, F1-score, receiver operating characteristic (ROC) curve and resource consumption. It also compares all the models with each other based on the above-mentioned metrics.

**Keywords:** Paraphrasing, Plagiarism, Paraphrase detection, Plagiarism detection, Paraphrase

## 1 Introduction

Paraphrasing is the procedure of rewriting a statement to change its form without changing the meaning of the original text. Many deep learning models have shown a promising performance in detecting semantic similarity between documents using word-embedding technique. Word embedding is a representation of document vocabulary which is capable of capturing the semantic context of the document as a whole, or the individual words in a document. The idea behind word embedding is assigning vector values to text, as the technique opens doors to various text evaluation and analysis approaches in the field of Linear Algebra.

The main objective of this paper is to review and compare the performance of various word-embedding based models (including some Deep Learning models) for semantic similarity detection which leads to following contributions: (a) Systematic outline of corpus-based word-embedding models, and (b) Elucidation of the best models,

preprocessing techniques and thresholds for plagiarism detection and the kind of corpus they work well with.

The motivation behind this research is to in-turn develop a framework which uses machine learning and statistics to measure semantic similarity between given documents and detect paraphrase plagiarism. The systematic study helps in cherry picking the mechanisms and hyperparameters best suited for a new framework.

This research compares five models such as Word2Vec [1], Doc2Vec [2], BERT [3], TF-IDF [4], FastText [5] on two publicly available corpora namely Quora Question Pairs and Plagiarized Short Answers (PSA) and in turn these models are evaluated in terms of accuracy, precision, recall and F1 score. Before applying the models, data preprocessing techniques, distance measures, and a threshold value for each model was set after careful experimentation on each corpus.

The paper is organized as follows: Section 2 briefly discusses existing work in the related field as well as specific work in the paraphrase detection area. Section 3 discusses approach followed in this paper, models implemented and the results derived from the experiments. Section 4 draws conclusion and discusses future scope.

## 2 Related Work

The use of computers in the field of Natural Language Processing (NLP) is a challenge due to the equivocacy of texts and passages. For example, the term ‘mango’ can be referred to as a fruit as well as a clothing brand. Hence, the semantic features of a text play a major role over and above its linguistic features.

A large number of established researchers have contributed to various plagiarism detection stages, [6]. For handling plagiarism and paraphrasing cases of higher complexities, [7] suggests that researchers should focus on linguistic, syntactic and most importantly semantic information of text rather than just on verbatim reproduction.

The following table (Table 1) summarizes various works and approaches in the field of paraphrase detection.

**Table 1.** Related work in paraphrase detection field

| Research work | Technique/ Algorithm | Dataset | Observations | Performance |
|---|---|---|---|---|
| [8] | Graph subsumption and isomorphism | Recognizing Textual Entailment (RTE) | Introduces a graph subsumption-based approach for paraphrase identification. In this, input sentences are mapped to corresponding graph structures and paraphrasing is evaluated based on graph isomorphism. | Accuracy: 65.90<br>Precision: 64.70<br>Recall: 70.00<br>F1-score: 67.24 |
| [9] | Neural Networks (NN) | (Microsoft Research Paraphrase) MSRP | Lexical features in association with syntactic, semantic and composite features are used to train a back | Accuracy: 75 and above. |

| | | | | |
|---|---|---|---|---|
| | | | propagation network, wherein extracted features are fed as input to the NN. | |
| [10] | Combination of deep neural network with keywords | MSRP; Sentences Involving Compositional Knowledge (SICK) [16] | Detects paraphrasing based on lexical and semantic similarity, measured by combining neural networks and keywords, and hence identifies the relation between the inputs in the vector space. | MSRP-<br>Accuracy: 80.2<br>F1-score: 86.4<br>SICK-<br>Accuracy: 87.1 |
| [11] | Textual Entailment | RTE | Paraphrase detection is viewed as a bidirectional textual entailment task. Word Sense Disambiguation is performed and using distant supervised learning technique, indirect facts are mined from the predicates | Accuracy: 78.6<br>F1-score: 86.3 |
| [12] | Logistic Regression (LR); SVM; Neural Networks | Quora Duplicate Question set | Compares various machine learning models after pre-processing and encoding the input and concludes Recurrent Neural Network (RNN) to be the most efficient algorithm for paraphrase identification task | Accuracies:<br>LR– 59.9<br>SVM – 62.5<br>Siamese NN – 63<br>Two Layer NN – 74<br>RNN – 80 |
| [13] | Word Alignment Information (WAI) | Quora Question Pairs (QQP) | Introduces a novel approach leveraging WAI to improve deep PI baseline model RE2 [17]. Employs two major schemes to test the performance, proving 'pre-training' the unlabeled in-domain data majorly improving performance of baseline model. | Accuracy on various schemes:<br>Embedding – 80.7<br>Multi-Task – 80.9<br>Pre-Train – 82.1<br>Hybrid (Pre-Train + Multi-Task) – 82.9 |
| [14] | Curriculum Learning | QQP; Large-scale Chinese Question Matching Corpus (LCQMC) | Estimates the effect of label noise on PI task and introduces an approach based on curriculum learning where: (a) a *loss-based noise metric* is developed to compute noise complexity of a sample, and (b) *similarity-based noise metric* classifies the paraphrase. | Accuracy:<br>QQP – 80.29<br>LCQMC – 80.93 |
| [15] | PIDG (Program Interaction Dependency Graph) | 12 Undergraduate programming assignments diversified with the help of SPPlagiarise [18] | Introduces a novel behavior-based *Source Code Plagiarism Detection Tool*, BPlag. Each program's behaviour is represented with the help of PIDG, and source code plagiarism is detected based on the similarity between the PID graphs. | Avg. Error Count at various % transformations: 4.3 |

Inspired by the recent successes of Neural Networks (NNs) in the fields of information retrieval and natural language processing, this paper experiments with various deep learning models to test semantic similarity detection power of the same and find out

various factors affecting their performance. The succeeding section discusses the approach adopted in this research paper.

# 3 Proposed Methodology

The goal of this paper is to measure and compare the efficiency of five word embedding models (Word2Vec, Doc2Vec, BERT, TF-IDF and FastText) in the task of plagiarism/paraphrase detection, on two publicly available corpora: Quora Question Pairs and Plagiarized Short Answers.

Appropriate pre-processing techniques are applied on both the corpora before starting with model testing. The models are then evaluated based on the most appropriate threshold values and distance measure, and the results are produced based on standard metrics - accuracy, precision, recall and F1-score. For each model tested against each corpus, ROC curve is generated for a deeper understanding of its performance.

## 3.1 Problem Definition

To formulate the task of paraphrase detection, the problem is defined as: Given two text inputs $T_1$ and $T_2$ such that $T_1 = \{T_1^1, T_1^2, T_1^3, \ldots, T_1^n\}$ and $T_2 = \{T_2^1, T_2^2, T_2^3, \ldots, T_2^n\}$. The task of paraphrase detection between the given inputs is formalized as a binary task where $L = \{0,1\}$ are the target labels such that if $T_1$ and $T_2$ are duplicate $L=1$ otherwise $L=0$ i.e. if not duplicate.

## 3.2 Corpora

The paper aims at comparing the listed models based on their semantic similarity detection. The following corpora, after taking into consideration their varied natures are tested on the models:

**Quora Question Pairs**

. In 2017, Quora Question Pairs[1], an international competition by Quora, was released to identify plagiarism and paraphrasing to be able to group similar/duplicate questions. The dataset contains genuine examples from the website with over 400,000 records.

**Plagiarised Short Answers**

. Plagiarised Short Answers [2](2009) is a corpus developed in a typical academic setting for the task of plagiarism identification in which four levels of plagiarism was committed. Since the input is classified based on a binary approach, four levels of paraphrasing are encoded to only two i.e., 0 (non-plagiarized) and 1 (plagiarized). The

[1] https://www.kaggle.com/c/quora-question-pairs/

[2] https://ir.shef.ac.uk/cloughie/resources/plagiarism_corpus.html

dataset contained 95 documents plagiarized at different levels against 5 original documents sourced from Wikipedia.
The two corpora differ a lot from each other in various aspects which is partially the reason why they were chosen. (a) While PSA is a clean dataset containing only textual information, QQP contains mathematical equations, abbreviations, slang, typos, etc. all of which can be accounted for as noise. (b) PSA is a smaller dataset as compared to QQP. (c) PSA also differs from QQP in the context that the constituent documents at question are larger in length in PSA as compared to QQP.

The different natures of both the datasets helped in evaluating and reviewing the word-embedding models more precisely.

Before experimenting with the models, the datasets need to be transformed into a standard form to maximize model performance. The following section discusses the various text preprocessing steps applied to both the datasets to achieve best results.

### 3.3 Text-Preprocessing

Before feeding the input to the models, the text corpora need to be pre-processed to eliminate noise and maximize the processing effect. The major pre-processing steps are briefly discussed below and represented in the flowchart (Fig 1) as well.

1. Each text input was transitioned from sensitive data to non-sensitive data by translating it to lower case and tokenizing it.
2. Each token obtained was checked against a dictionary of stop-words obtained from *nltk* library in python. If the word matched against the dictionary, it was dropped to avoid unnecessary processing, as stop-words rarely have any role in plagiarism.
3. Three alternatives of number manipulation were tested,
   a. Leaving them,
   b. Replacing them with <NAN> and
   c. Deleting them from the text.
   Best results were achieved once they were removed altogether.
4. Stemming [19] was applied to original tokens to reduce them to their word stem or word root, but the results proved to be better without this step, hence was not carried forward.
5. Lemmatization [20] was applied by using *WordNet* Lemmatizer [21]. Lemmatization helps in retaining the context of text or achieving its base form. For e.g., New and York are individual tokens which when lemmatized, are treated as a single entity i.e., New York and hence the context of the text is retained. The results proved to be slightly better when lemmatization was applied.
6. The tokens were then combined to form original text.
7. The original text was checked for punctuation marks and contractions. Punctuation marks were straight away eliminated while contractions like can't, she'll, I've, etc., were expanded to their original form – here, cannot, she will, I have respectively.
8. The whole corpus is then searched for duplicate entries which are subsequently removed to avoid overhead. The total number of duplicates were 353 which accounted for only about 0.9% of the whole dataset and hence didn't lead to any unbalancing.

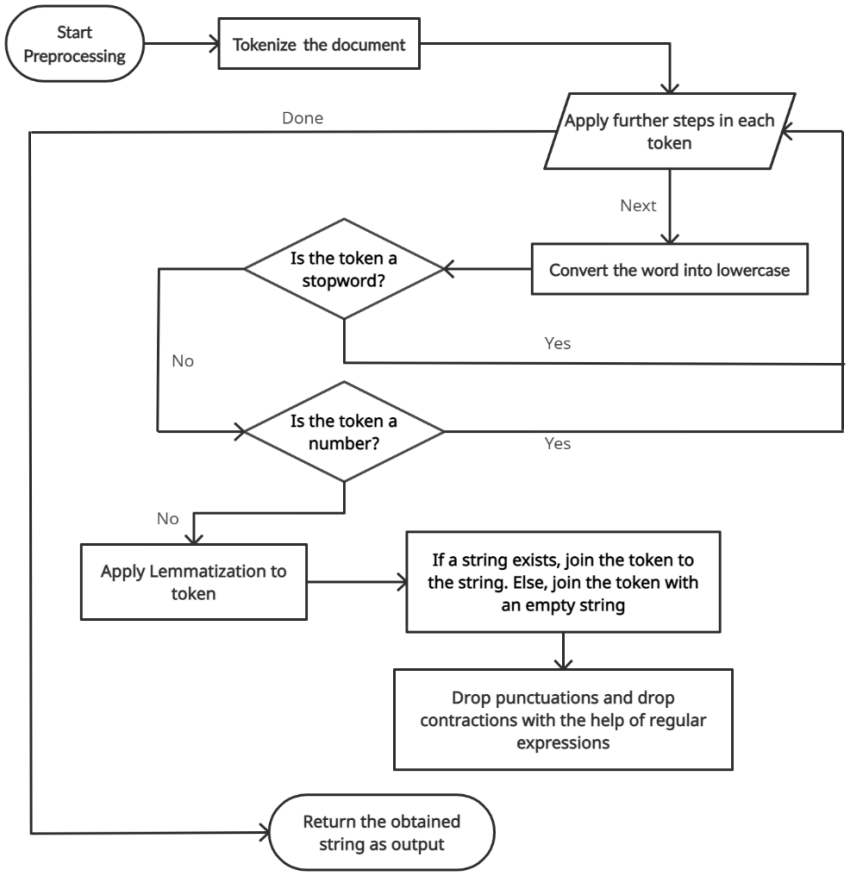


**Fig. 1.** Flowchart of text preprocessing steps applied to each corpus

The above steps are applied to each dataset. No additional preprocessing is done on any particular dataset to draw a fair result when tested with the models.

Following section discusses the various models which are applied on this preprocessed data to test semantic similarity of the documents through word embeddings.

### 3.4 Models Referred

. Machine Learning and almost all Deep Learning models are incapable of processing text in its raw form. One of the first and most popular approach to measure the semantic similarity between texts was Vector Space Model [22], which introduced space density computations on text such that each entity (such as characters in words, words in sentences, sentences in documents and documents in dataset) could be

represented in an n-dimensional space as a vector. Hence, the proximity of two entities in the space infer their semantic similarity.

Different models employ different strategies to utilize VSM for word embedding of input. A brief description of the five models taken on for research is given below:

**TF-IDF**

. The paper proceeds with the oldest word embedding model i.e., Term Frequency-Inverse Document Frequency which is a statistical frequency based embedding strategy developed by [23], and measures the significance of a term among the corpus of documents. The words in a document are given different weights through the following equation:

$$w_{ij} = tf_{ij} \times \log (N/df_i') \quad (1)$$

where $tf_{ij}$ = number of occurrences of term i in document j, $df_i$' = number of documents containing the term i, and N = number of documents.

**Word2Vec**

. Word2Vec is a deep learning model which is actually a combination of two strategies – Continuous Bag of Words (CBOW) and skip-grams. Both of these techniques learn weights of each input word which are thus taken as the final word vector representations.

The model creates vectors which are distributed numerical representations of word features such as the context of individual words. With enough data, usage and contexts, Word2Vec can make accurate guesses about a word's synonyms based on its previous appearances (Fig 2)

```
model = Word2Vec(questions_labeled, min_count=2, size=50, workers=4 )
model.wv.most_similar('book')
```

```
[('books', 0.8578062653541565),
 ('novel', 0.7103374004364014),
 ('novels', 0.6823523044586182),
 ('poem', 0.6142452359199524),
 ('ebook', 0.6063246130943298),
 ('textbook', 0.6029399633407593),
 ('journal', 0.5949798822402954),
 ('biography', 0.5928425788879395),
 ('pandey', 0.5822772979736328),
 ('congenitally', 0.5763487219810486)]
```

**Fig. 2.** Using Word2Vec to extract the most similar value of a given word based on nearest vector values plotted in n-dimensional space.

Fig 3. depicts the distribution of a small set of computed vectors in an n-dimensional space.

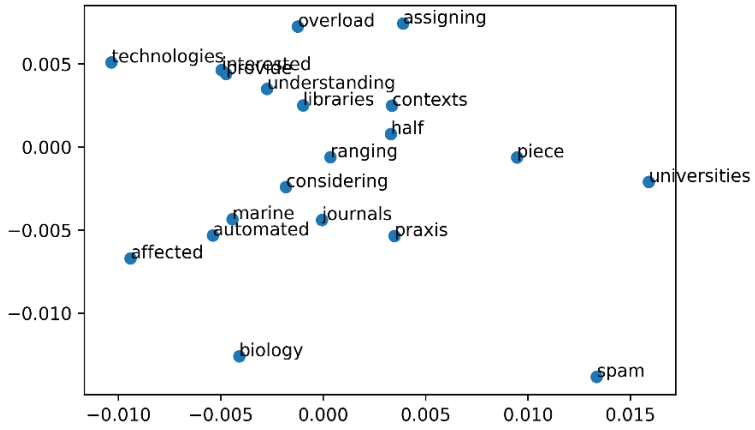


**Fig. 3.** Distribution of vectors computed by Word2Vec model in an n-dimensional space

**BERT**

. Sent2Vec is a sentence to vector strategy where distributed representations of sentences are achieved. Rather than extracting the semantic meaning of a word, the whole sentence is taken into consideration. It can be thought of as an extension of word2vec to sentences whereby, vectors are the average of source word embeddings.

BERTSimilarity library from pandas is used as sentence embedding technique in this research which employs forward pass of BERT model to compute vectors for sentences.

**Doc2Vec**

. Using this strategy, the document can be represented as vectors by using paragraph vector algorithm introduced by Mikolov, et al. in [2]. The model basically remembers the context of words encountered and hence the whole document can be plotted as a vector based on its semantic meaning.

**FastText**

. FastText is a powerful word embedding model introduced by Facebook where in a word is assumed to be formed by n-grams of characters, example, rainy can be represented as [rainy, rain, ainy], [rai, ain, iny]. It is particularly efficient over traditional approaches in the sense that, it accounts for rarer words and can give vector representations to even the words absent from the dictionary.

All the above models have something in common: generating vector values for textual data. Raw textual data is incapable of extensive experimentation as majority of the existing algorithms work on numerical data. The advantage with extracted vector values is that it allows putting the data through various concepts of Linear Algebra, which opens doors to great level of experimentation and findings. The following section discusses how the generated word embeddings can be used to analyze semantic similarity between documents.

### 3.5 Distance computation between word embeddings to detect plagiarism

After generating word embedding for the corpora based on various models mentioned above, a distance measure was chosen to calculate the actual distance between vectors in the n-dimensional space. This paper compares the vectors using the cosine distance measure which can be obtained from cosine similarity. The cosine similarity between two entities T1 and T2 can be calculated as:

$$similarity\ (T_1, T_2) = \cos(\Theta) = (T_1 \cdot T_2) / ( ||T_1|| \times ||T_2|| ) \quad (2)$$

The cosine distance can be calculated from similarity such that,

$$distance\ (T_1, T_2) = 1 - similarity\ (T_1, T_2) = 1 - \cos(\Theta) \quad (3)$$

The distance between the two entities is checked against a threshold value. If the score is greater than or equal to threshold, the entities are reported as plagiarized.

Since cosine distance is used as a metric, it is a given that the result of comparison will always be in the range [0,1]. Thus, the threshold always lies in the range [0,1].

The threshold value chosen varies according to the model into consideration and context of the given dataset. Hence, it is not a given that the same threshold value will produce the most accurate result for a model over any corpus and should be carefully chosen after experimentation.

The different threshold values which produced the most efficient results for chosen datasets are presented in the table below (Table 2).

**Table 2.** Threshold values to classify plagiarized records in each model corresponding to the corpus

| | Quora Question Pairs | Plagiarized Short Answers |
|---|---|---|
| TF-IDF | 0.52 | 0.54 |
| Word2Vec | 0.90 | 0.98 |
| BERT | 0.89 | 0.96 |
| Doc2Vec | 0.94 | 0.93 |
| FastText | 0.92 | 0.92 |

### 3.6 Evaluation and Results

Standard measures are used in this paper to calculate the performance of each model. The measures include accuracy, precision, recall, and F1-score and can be calculated with the help of *sklearn* library in python. The library gives the scores under two major categories: macro-avg and weighted-avg. We have considered weighted-avg in our evaluation to accommodate the size positive and negative samples.

The following table (Table 3) summarizes the values obtained of mentioned metrics against each model and corpus.

**Table 3.** Performance of models at their best thresholds

| | Accuracy | | Precision | | Recall | | F1-Score | |
|---|---|---|---|---|---|---|---|---|
| | QQP | PSA | QQP | PSA | QQP | PSA | QQP | PSA |
| TF-IDF | 0.64 | 0.79 | 0.65 | 0.86 | 0.64 | 0.79 | 0.64 | 0.80 |
| Word2Vec | 0.64 | 0.73 | 0.63 | 0.74 | 0.64 | 0.74 | 0.64 | 0.73 |
| BERT | 0.66 | 0.77 | 0.66 | 0.83 | 0.66 | 0.77 | 0.66 | 0.77 |
| Doc2Vec | 0.65 | 0.74 | 0.66 | 0.72 | 0.65 | 0.79 | 0.65 | 0.753 |
| FastText | **0.69** | **0.93** | **0.70** | **0.92** | **0.69** | **0.92** | **0.69** | **0.92** |

Table 2 reveals that the threshold values for both the corpora against a particular model vary diversely. Table 3 gives the performance summary of all the tested models. The behaviour is dramatically different when the evaluation metrics for both the corpora are compared against each other for all the models.

Both QQP and PSA datasets were made to undergo the same pre-processing steps for a fair comparison but QQP is a much larger dataset compared to PSA. Hence, it is lead to believe that the noise factor (like unaccounted abbreviations, orthographical errors, etc.) plays a major role in the performance of a model for a particular corpus. The derivation comes from the following two reasons:

- Deep Learning models should ideally perform better when the size of dataset is large to train their neural network. Therefore, QQP should have worked well with all the models because of its size. But the performance of models, in contrast to the expectation has greatly worsened in QQP dataset, accounting that noise has a role to play.
- Since FastText is noise tolerant to some extent as explained in Section 3.4, it should be able to account for some faults in the dataset. The evaluation metrics prove this point, as FastText performs significantly better on both corpora.

Another probable but indefinite reason for the poor performance of DL models on QQP dataset can be the presence of label noise. Label noise is extensively noticed in paraphrase identification datasets due to automatic labeling or non-expert labeling and it can severely impact the outcome of deep learning models as mentioned in [45]. QQP has undergone human labeling to classify a pair of questions as paraphrased or non-paraphrased.

The presence of label noise however, could not be confirmed in datasets due to shortage of time during implementation.

The ROC curves for various models on each dataset are depicted in Fig 4 and Fig 5 to help assess the performance of the models further. As mentioned in [24], a perfect diagnostic test would constitute an ROC curve that is almost vertical from (0,0) to (0,1) and then horizontal to (1,1).

It is evident that FastText exhibits the best performance among all the listed models, with reference to Table 3 and Fig 5.

Even though BERT performs slightly better than Word2Vec it has considerably higher hardware and processing requirements i.e. consumes more resources and time.

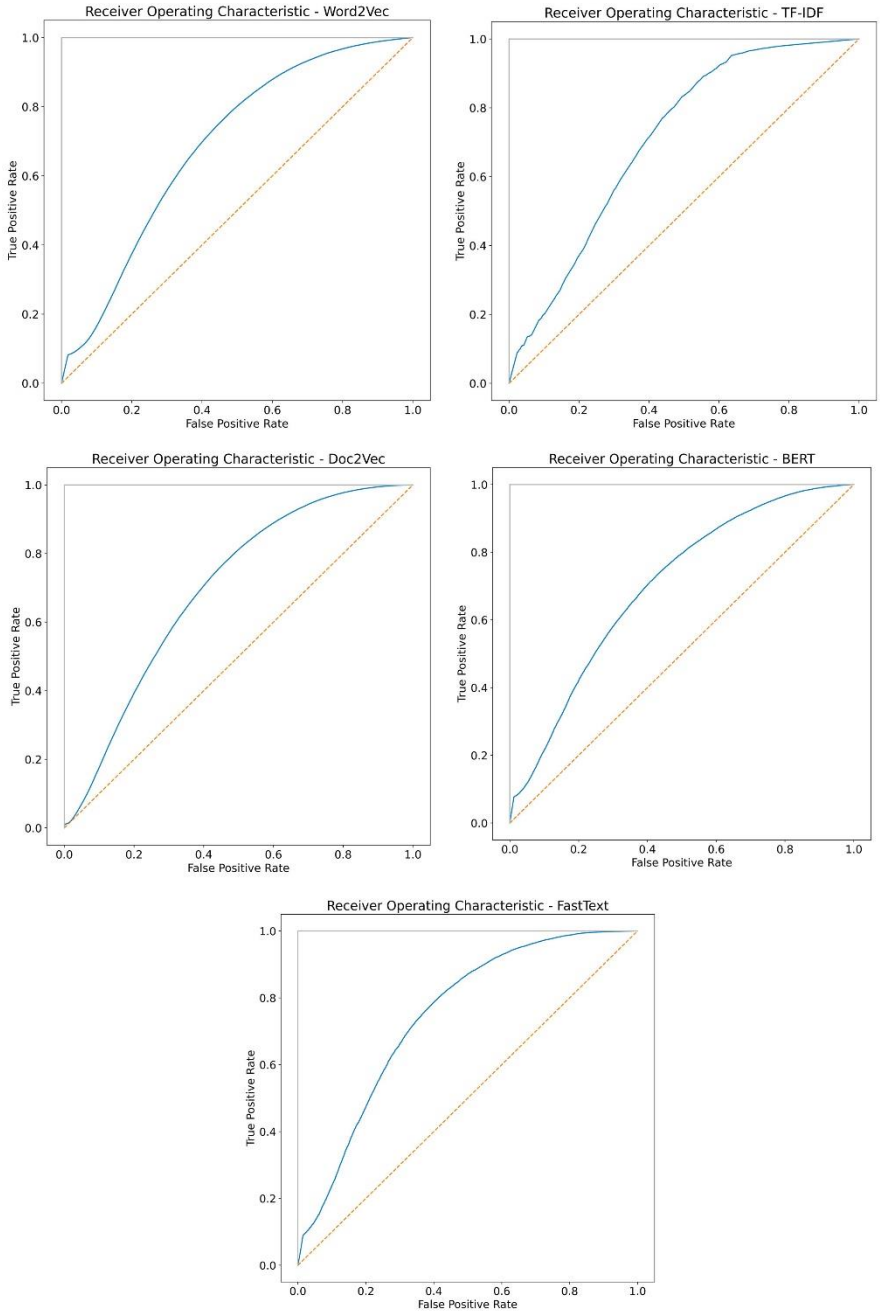


**Fig. 4.** ROC Curves for various models on QQP dataset (left to right, top to bottom: Word2Vec, TF-IDF, Doc2Vec, BERT, FastText)

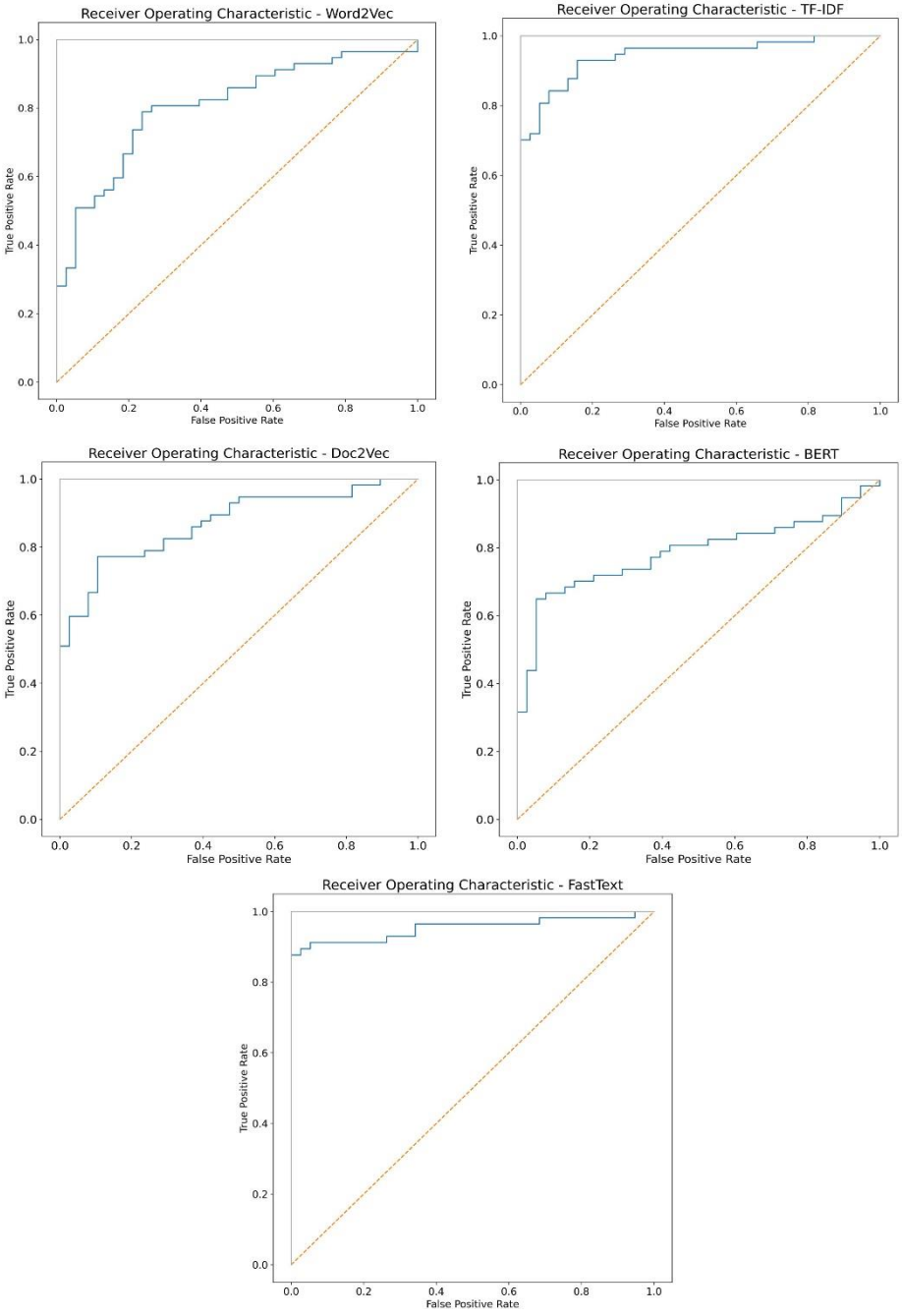


**Fig. 5.** ROC Curves for various models on PSA dataset (left to right, top to bottom: Word2Vec, TF-IDF, Doc2Vec, BERT, FastText)

Doc2Vec has better performance over BERT as well as Word2Vec, both in terms of evaluation metrics and resource consumption. Contrary to expectations, TF-IDF performs at par with Doc2Vec, Word2Vec and BERT.

## 4 Conclusion and Future Scope

This paper proposes comparison of various corpus based deep learning as well as statistical word embedding models, specifically TF-IDF, Word2Vec, Doc2Vec, BERT and FastText. The models are tested on two publicly available corpora namely – Quora Question Pairs and Plagiarized Short Answers. This research work helps in understanding the performance of various word-embedding techniques on the mentioned corpora and effect of variations of various parameters like threshold, hyperparameters, distance measures, preprocessing steps, on the evaluation metrics.

It is concluded that FastText produces the most accurate and efficient results and BERT is the most resource demanding model. Based on experimental results, it can also be deduced that standalone Deep Learning models are not quite sufficient for plagiarism detection, and can be improved using additional mechanisms, for instance, negative sampling, dissimilarity measures, label noise elimination etc. The paper also proves that TF-IDF has shown unexpected performance and hence, probably holds the capability to produce even better results, either in combination with another well-suited DL model or with modifications to account for variations, like negative sampling.

In future, this research can be extended on other state of art models like GloVe, ELMO, USE, etc. LSI can be considered as well for comparison of traditional models with state-of-the-art models. Another gap in this research is lack of testing on a corpus which is mid-way between PSA and QQP i.e., a dataset that is significantly larger as compared to PSA but has significantly lower noise when compared to QQP. Another possible variation of the dataset can consider the length of constituent documents to measure the trade-off between evaluation metrics and length. In addition to this, the results show great strength in TF-IDF model and the said can be improved.

## References

1. T. Mikolov, et al., “Distributed representations ofwords and phrases and their compositionality,” Adv. Neural Inf. Process. Syst., pp. 1–9, 2013.
2. Q. Le and T. Mikolov, “Distributed representations of sentences and documents,” 31st Int. Conf. Mach. Learn. ICML 2014, vol. 4, pp. 2931–2939, 2014.
3. Devlin, Jacob, et al., “BERT: Pre-training of deep bidirectional transformers for language understanding,” NAACL HLT 2019 vol. 1, pp. 4171–4186, 2019.
4. J. Ramos, “Using TF-IDF to Determine Word Relevance in Document Queries,” Proc. first Instr. Conf. Mach. Learn., vol. 242, no. 1, pp. 29–48, 2003.
5. A. Joulin, E. Grave, et al., “Bag of tricks for efficient text classification,” 15th Conf. Eur. Chapter Assoc. Comput. Linguist. EACL 2017 - Proc. Conf., vol. 2, pp. 427–431, 2017
6. Alzahrani, N. Salim, et al., “Understanding plagiarism linguistic patterns, textual features, and detection methods,” IEEE Trans. Syst. Man Cybern., vol. 42, no. 2, pp. 133–149, 2012.

7. V. M.K and K. K, “A Survey on Similarity Measures in Text Mining,” Mach. Learn. Appl. An Int. J., vol. 3, no. 1, pp. 19–28, 2016.
8. V. Rus, P. M. Mccarthy, M. C. Lintean, D. S. Mcnamara, and A. C. Graesser, “Paraphrase Identification with Lexico-Syntactic Graph Subsumption,” pp. 201–206, 2008.
9. Rajkumar, “Paraphrase Recognition using Neural Network Classification,” 1. 29. 43–48, 2010.
10. X. Wang and B. Xu, “Paraphrase Recognition via Combination of Neural Classifier and Keywords,” 2018.
11. S. Seethamol and K. Manju, “Paraphrase identification using textual entailment recognition,” ICICICT 2017, vol. 2018-Janua, pp. 1071–1074, 2018.
12. E. Hunt, R. Janamsetty, et al., “Machine Learning Models for Paraphrase Identification and its Applications on Plagiarism Detection,” 2019 IEEE Int. Conf. Big Knowl., pp. 97–104, 2019.
13. Wang, B. & Wang, L., “Enhancing Deep Paraphrase Identification via Leveraging Word Alignment Information”, CNCERT/CC. pp: 7843–7847, 2021
14. B. Li, T. Liu, B. Wang, et al., "Label Noise Robust Curriculum for Deep Paraphrase Identification," 2020 International Joint Conference on Neural Networks (IJCNN), pp. 1-8, 2020
15. H. Cheers, Y. Lin and S. P. Smith, "Academic Source Code Plagiarism Detection by Measuring Program Behavioral Similarity," in IEEE Access, vol. 9, pp. 50391-50412, 2021.
16. M. Marelli, et al., “A SICK cure for the evaluation of compositional distributional semantic models,” Proc. 9th Int. Conf. Lang. Resour. Eval. Lr. 2014, no. May, pp. 216–223, 2014.
17. Runqi Yang, et al., “Simple and effective text matching with richer alignment features,” in Proceedings of the 57th Annual Meeting of the ACL, pp. 4699–4709, 2019
18. H. Cheers, Y. Lin, and S. P. Smith, “Spplagiarise: A tool for generating simulated semantics-preserving plagiarism of java source code,” in 2019 IEEE 10th ICSESS, 2019, pp. 617–622.
19. M. F. Porter, “An algorithm for suffix stripping,” Program, vol. 14, no. 3, pp. 130–137, 1980.
20. C. Callison-burch, “Syntactic Constraints on Paraphrases Extracted from Parallel Corpora,” no. October, pp. 196–205, 2008.
21. Fellbaum, Christiane. "WordNet." In Theory and applications of ontology: computer applications, pp. 231-243. Springer, Dordrecht, 2010.
22. G. Salton, A. Wong, and C. S. Yang, “A Vector Space Model for Automatic Indexing,” Commun. ACM, vol. 18, no. 11, pp. 613–620, 1975.
23. Salton, Gerard, and Christopher Buckley. "Term-weighting approaches in automatic text retrieval." Information processing & management 24, no. 5, pp: 513-523, 1988.
24. NCSS statistical software, “Comparing Two ROC Curves - paired design,” pp. 1–14, 1997.
25. Anjali, A. Tripathi, S. Gupta and S. Singh Yadav, "Plagiarism Detector using Long Term Common Tracking," 2021 ICACITE, pp. 184-189, 2021.
26. S. Zouaoui and K. Rezeg, “Multi-Agents Indexing System (MAIS) for Plagiarism Detection,” J. King Saud Univ. - Comput. Inf. Sci., no. xxxx, 2020.
27. S. Xu, S. E, and Y. Xiang, “Enhanced attentive convolutional neural networks for sentence pair modeling,” *Expert Syst. Appl.*, vol. 151, p. 113384, 2020.
28. J. Zhan and B. Dahal, “Using deep learning for short text understanding,” Journal of Big Data, vol. 4, no. 34, pp. 1–15, 2017.
29. D. Sakamoto and K. Tsuda, “A detection method for plagiarism reports of students,” *Procedia Comput. Sci.*, vol. 159, pp. 1329–1338, 2019.
30. Ruder, Sebastian, Ivan Vulić, and Anders Søgaard. "A survey of cross-lingual word embedding models." Journal of Artificial Intelligence Research 65, pp: 569-631, 2019.
31. I., Mohamed & Gomaa, Wael. Exploring the Recent Trends of Paraphrase Detection. International Journal of Computer Applications. 182. 1-5. 2019.
32. J. Zhan and B. Dahal, “Using deep learning for short text understanding,” *J. Big Data*, vol. 4, no. 1, 2017.