Plagiarism Detection in Tamil Text using Word Embedding

M A Munzir and E Y A Charles
Faculty of Science, University of Jaffna
munzirahmadh@gmail.com, charles.ey@univ.jfn.ac.lk

Abstract

- This poster presents a research work on plagiarism detection in Tamil text. Drawbacks and inefficiency of existing language independent plagiarism detection tools and lack of adequate research for Tamil plagiarism detection motivated this research work.
- Proposed method focuses on developing a word embedding based approach for plagiarism detection in Tamil text incorporating Natural Language Processing (NLP) techniques.
- Proposed method was able to detect plagiarism using fastText with an accuracy of 96.66% and capable to capture semantic and syntactic similarities between documents.

Introduction

- According to the Oxford dictionary, “Plagiarism is presenting someone else’s work or ideas as your own, with or without their consent, by incorporating it into your work without full acknowledgment”.
- Word embedding is a deep learning method which represents text as a vector of values where words with similar meanings has similar representation.
- Word2vec is word embedding developed by a group of researchers headed by Thomas Milkov at Google.
- FastText is another popular word embedding developed by Facebook. This model is constructed using n-grams of text hence can provide representation for words that were not used to build embedding model.
- Both word2vec and fastText have the same goal: to learn vector representations of words. Word2vec model for a language can be built using the continuous bag-of-words (CBOW) approach or continuous skip-gram. But fastText is built using a more granular level approach using character n-grams. Here words are represented by the sum of the character n-gram vectors.

Objective

- The main objective of this research project is to build a NLP System to detect plagiarism in Tamil text. There are no language-dependent plagiarism detection systems available for Tamil text. This research intend to develop a plagiarism detection system using a deep learning approach.

Conclusion

- Embedding models are an easier and simplest way to identify the similarity of sentences for plagiarism detection. Proposed approach got a better accuracy (96.66%) in detecting plagiarism in Tamil text. Further precision of the proposed model is 0.9473, recall is 1, and F-measure is 0.9729. This was achieved with 300 dimension word embedding vectors.

Methodology

Corpus for Embedding model building

- Wikipedia Articles, Tamil Murasu news, Dinamalar news and some common sentences that are publicly available. Collected documents contained lots of noises such as HTML syntax and punctuations.

Embedding model building

- As the first step, the collected documents were combined as a single corpus or single file and cleaned.
- In the cleaning step, basic pre-processing steps such as tokenization, punctuation removal and stop word removal are applied to achieve maximum possible accuracy. Here tokenization is used to break down the text into small parts or words.
- After preprocessing, word2vec continuous bag words model, word2vec skip-gram model, and fastText model with the dimensions 100, 200, 300 were built. Training loss and perplexity were used as the performance measure of the model.
- After many trials window size have been selected as 10, minimum word count as 1, down sampling as 0.003 and number of as 100.

Plagiarism Detection

- As the first step, words in a source document and documents to be compared are represented as vectors using an embedding model.
- Those vectors are used to generate the vectors for the sentences. Using the sentence vectors, source document is compared with other documents using Cosine, Soft-Cosine similarity and Euclidian distance.
- As a measure for similarity of documents a threshold value based on average of Cosine and Soft-Cosine similarities and a threshold value of Euclidean distance are used. Based on the threshold values the source document is determined as plagiarized or not.

Data set for Plagiarism Detection

- From the dataset that is used for model training, 15 documents were randomly selected and the sentences were plagiarized by a selected group of students. In order to reduce the error probability, the plagiarized text were checked by a Tamil language expert. In addition to that the sentences were labeled as similar or not.

Methodology

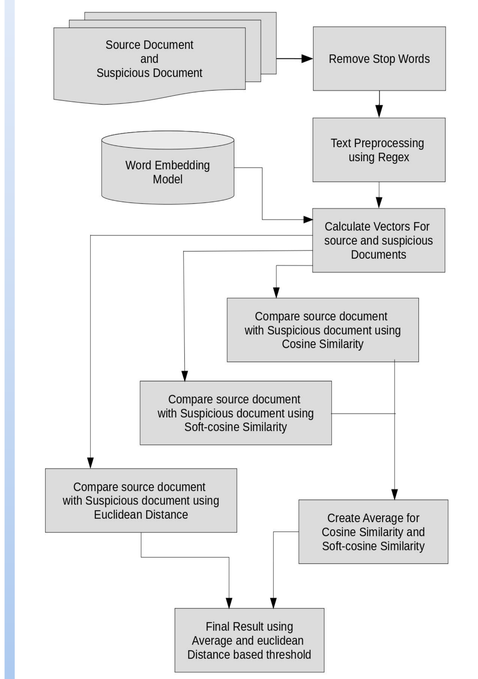

Figure 1 : Proposed approach of research work

Reference

- Thomas Milkov, Kai Chen, Greg Corrado, Jeffrey Dean, “Efficient Estimation of Word Representation in Vector Space”. In International Conference on Learning Representations, 2013
- Bojanowski, P., Grave, E., Joulin, A., & Milkov, T, “Enriching Word Vectors with Sub word Information”, Transections of the Association for computational Linguistics, 5, 135-146. 2013.
- Dima Suleiman, Arafat Awajan, Nailah Al-Madi, “Deep Learning Based Technique for Plagiarism Detection in Arabic Texts”, International Research Conference on New Trends in Computer Science, 2017.
- Tharuka KasthuriArachi, Charles E Y A, “Deep Learning Approach to Detect Plagiarism in Sinhala Text”, International Conference on Information Intelligence and Systems, 2019.

Results and Discussion

- In our research work we used confusion matrix to find the accuracy of the model. Along with that to ensure the model is working with better precision, recall and F-measure is used for plagiarism detection.
- In this research work we obtained 96.66% as accuracy for fastText embedding with smooth inverse frequency and 94% of accuracy for without smooth inverse frequency when using dimension 300. Here fastText works as best embedding model for our research.

Accuracy (%)									
Threshold Value	**CBOW 100**	**CBOW 200**	**CBOW 300**	**SK 100**	**SK 200**	**SK 300**	**FT 100**	**FT 200**	**FT 300**
0.450	88.00	89.33	92.00	85.33	86.66	86.00	84.00	84.66	84.00
0.475	88.66	92.00	92.00	86.66	89.33	88.00	84.00	84.66	84.00
0.500	89.33	92.00	91.33	87.33	90.66	89.33	86.00	87.33	86.66
0.525	91.33	**92.66**	**93.33**	89.33	90.66	90.66	86.00	87.33	86.66
0.550	**92.66**	92.00	90.66	90.66	90.66	90.66	87.33	90.00	88.00
0.575	**92.66**	90.00	87.33	91.33	90.66	91.33	87.33	90.00	90.00
0.600	**92.66**	86.66	84.00	91.33	**92.66**	92.00	90.00	90.66	94.00
0.625	86.66	83.33	80.00	**92.00**	89.33	**94.00**	90.00	92.00	**96.66**
0.650	84.00	77.33	74.00	91.00	84.66	92.66	90.66	**93.33**	92.00
0.675	80.00	72.66	71.33	88.00	82.00	90.66	**91.33**	90.66	88.00
0.700	76.00	72.66	70.00	87.00	82.00	89.33	91.33	89.33	88.00

Table 1 : Accuracy values obtained from the fastText model with 300 dimension using smooth inverse frequency

Accuracy (%)									
Threshold Value	**CBOW 100**	**CBOW 200**	**CBOW 300**	**SK 100**	**SK 200**	**SK 300**	**FT 100**	**FT 200**	**FT 300**
0.450	88.00	89.33	90.66	85.33	86.66	86.67	84.00	84.66	84.67
0.475	88.66	92.00	91.33	86.66	89.33	88.00	84.00	84.66	84.00
0.500	89.33	92.00	91.33	87.33	90.66	89.33	86.00	85.33	85.33
0.525	91.33	**92.66**	92.00	89.33	90.66	90.66	86.00	87.33	86.66
0.550	**92.66**	92.00	**92.66**	90.66	**92.00**	91.33	87.33	90.00	88.00
0.575	**92.66**	90.00	89.33	91.33	91.33	91.33	87.33	90.00	90.00
0.600	**92.66**	86.66	87.33	**92.00**	90.00	92.00	88.66	90.66	92.00
0.625	86.66	83.33	84.66	91.33	86.66	**93.33**	**90.66**	92.00	**94.00**
0.650	84.00	77.33	80.00	91.00	84.66	90.00	**90.66**	**93.33**	92.00
0.675	80.00	72.66	74.00	87.00	77.33	88.00	**90.66**	91.33	88.00
0.700	76.00	72.66	71.33	86.00	74.00	86.00	90.00	89.33	88.00

Table 2 : Accuracy values obtained from the fastText model with 300 dimension without using smooth inverse frequency

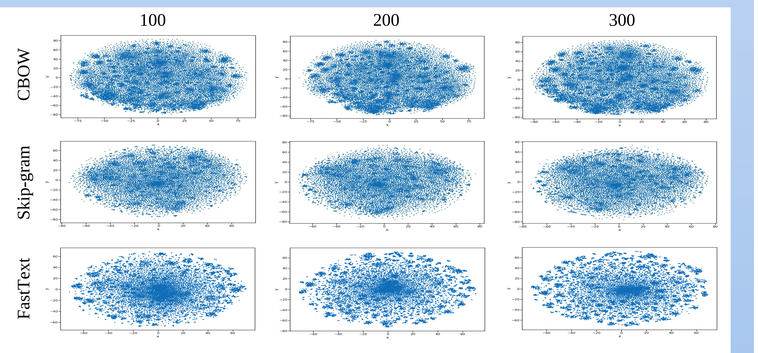

Figure 2 : Illustration of models. Model contains 1MB corpus when plotting the models. This representation is how each model distributed as vectors in a n-D to 2-D model.