Common Text Detection

Anamitra Maji
Dept. of Computer Science and Engineering
Indraprastha Institute of Information Technology,
Delhi
anamitra19112@iiitd.ac.in

Akanksha Dewangan
Dept. of Computer Science and Engineering
Indraprastha Institute of Information Technology,
Delhi
akanksha19049@iiitd.ac.in

ABSTRACT

This project is about finding similarity in sentences within a set of documents. In brute force approach which says " STRING COMPARISON " in which we need to compare each and every sentence present in all the documents in the dataset whose time complexity is in polynomial order of 2 or more which is not efficient when there are large number of documents, So we came up with an algorithm which used vector representation of sentences and clustering to optimize time complexity. In the optimised algorithm we did feature extraction using emebbeders like " word2vec, glove " , along with statistics like " average, standard deviation, variance, mean, median " to get vector representation of sentence and then applied k-mean clustering. We created the new documents from input documents. To do that we randomly copied sentences from the input documents and put it in a new generated document. Along with that we keep the information about which line of the original document is copied in the newly generated document. In this way we created the ground truth.
Here we tried different embedder-statistics combination to observe at which point of x-axis(the total number of documents) our proposed algorithm started beating brute force method. Glove with mean showed the above reflection in minimum number of documents whereas word2vec with median took maximum numbers of documents.

1. INTRODUCTION

Nowadays plagiarism is a serious problem. Mostly plague is explained as supposingly as a paragraph or some sentences are copied from somewhere happened among university students, also peoples do copy paste from the internet there activity need to be detected. There are many tools available in the market which are using various techniques for detecting sentence similarity and identifying percentage of similarity in documents. For common text detection we need to apply brute force technique where each and every line of original documents are compared with new text documents for finding similarity whose time complexity is in order of two that is too slow. So we need an optimized approach for detecting similarity of sentences.

Plagiarism detectors usually characterize unstructured documents using various categories of textual features such as lexical, syntactic, and semantic.[1] Literal plagiarism includes copy–paste operations and is usually easy to detect. More sophisticated forms of plagiarism may involve translation, summarization, and paraphrasing and are more difficult to recognize.[2]
Copy detection task approach in the well known system is SCAM(Stanford Copy Analysis Mechanism)(Shivakumar and Garcia Molina, 1996) has two web based objectives: first to check the internet for copyright and another for filter out duplicates and near duplicates in information retrieval. To do this they process several million web pages, then take out a set of chunks. A chunk is a set of strings of words based on non-overlapping chunks that reduces storage requirements but may lose much useful information. If two chunks match in all except one word they will not match at all. This problem of address by Broder(1998) the task of clustering 30,000,000 documents into groups those can be closely resemble to each other, concept based on matching sequence of words they say "shingles".[4]
After surveying various techniques we implemented different feature extraction methods in combination of embedders summarization after that applied clustering. Then taking out the distance of all the sentence from cluster centroid and take out similar distance sentences from centroid results in reduced sentences for comparison so we reduce time complexity

2. PROPOSED METHOD

The methodology which is followed in our algorithm along with data generation steps. In 'Algorithm Flowchart' section breif about all the methodology such as sentence documentation mapping then applied two type of embedders like word2vec glove along with 10 different type of summarization technique which applied in sentences like avergae with variance, avergae with standard deviation, median mean. After embedding we do clustering over that to find out maximum match as per distance from centroid then did sentence vector comparision.

2.1 Data-preparation flowchart

In this section brief explanation about Data preparation using original documents. Here we follow steps as shown in flowchart in figure 2 1.

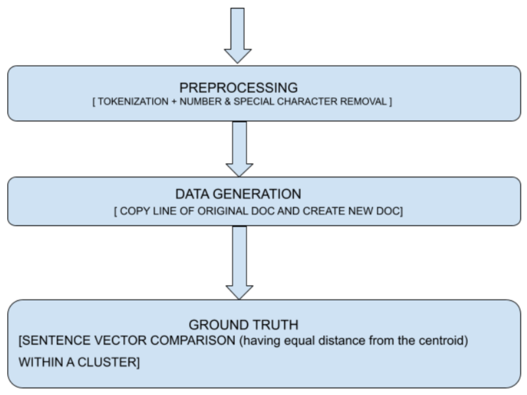

PREPROCESSING
[TOKENIZATION + NUMBER & SPECIAL CHARACTER REMOVAL]
DATA GENERATION
[COPY LINE OF ORIGINAL DOC AND CREATE NEW DOC]
GROUND TRUTH
[SENTENCE VECTOR COMPARISON (having equal distance from the centroid)
WITHIN A CLUSTER]

Figure 1: Data generation flowchart

2.1.1 Pre-processing

1. Sentence tokenization documentwise.
2. Tokenize the sentences.
3. Remove numbers along with extra special characters.
4. Take those tokenized sentences whose length is greater than equal to 8.

2.1.2 Data Generation

Generated new documents by randomly copying sentences from the input document and put it in a new document. Along with that we kept the information of the new document about sentence from which it is copied in the new document, say ground truth for the new document.

2.1.3 Ground Truth

In this we created the ground truth of a document by keeping information about sentences. Like which sentence belongs to which document and what is their line number in which that sentence is present. It was stored in '.json' format which uses further to evaluate the correctness of sentence matching.

2.1.4 Vectorization

It is the process in which sentences are transformed into numerical values, Where each value in vector represent its importance in the document.

2.2 Algorithm Flowchart

In algorithm we followed the flowchart as in figure 1 2 where intially we load the generated data which was prepared earlier by copying sentences from original document and prepare new document by keeping information of which sentence of which document is copied in which line of which new document that's came up as a ground truth.

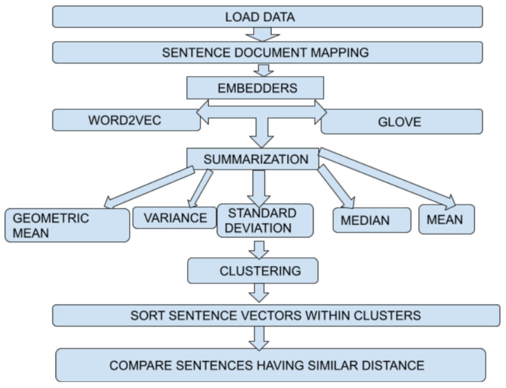


Figure 2: Optimized algorithm flowchart

2.2.1 Embedders

" Word embedding is a method that represents words of sentences into numerical values and represent those values into vectors which are of fixed size of vocab corpus."

Learning in this method is done by either joining with a neural network model on a particular task like document classification, in an unsupervised algorithm using document statistics. This section uses two techniques used to word embedding from text data.

- Word2vec-
  "Word2Vec is a statistical method for efficiently learning a standalone word embedding from a text corpus."

  Text corpus is input to word2vec and Feature vector is the outcome of word2vector embedding, which is a set of vectors having numerical value that represents the word importance in a document.

- Glove-
  "The Global Vectors for Word Representation, or

GloVe, algorithm is an extension to the word2vec method for efficiently learning word vectors."
Glove is an unsupervised algorithm which is used to obtain vector representations for set of words. This is done by mapping words This is achieved by mapping words into a meaningful space where the distance between words is related to semantic similarity.

2.2.2 Summarization

We have word vectors for each word present in sentences. Now using the following statistics(average, median, average etc) we built the final sentence vector from word vectors.

- average with standard deviation:
  Took average variance of word vectors and apply four operators with them to get the final sentence vector. four different operators include following:
  - Minus
  - concatenation
  - Multiplication
  - plus/addition
- standard deviation:
  Get the final sentence vector by calculating the standard deviation of the word vectors.
- variance:
  Get the final sentence vector by calculating the variance of the word vectors.
- median:
  Get the final sentence vector by calculating the median of the word vectors.
- root mean square:
  root mean square of word vectors to get the final sentence vector.
- Geometric Mean:
  Geometric Mean of word vectors then got our final sentence vector.
- Harmonic Mean:
  To get the final sentence vector we had using Harmonic Mean.

2.2.3 Clustering

"Clustering is an Unsupervised Learning algorithm that groups data samples into k clusters." In k-means clustering 'k' represents the number of clusters in the algorithm. Mathematically euclidean distance between the centroid would be minimum in a cluster. Basically clustering is a method of putting similar catalogs together in the same group. In terms of documents we can say clustering form structure where similar documents are grouped in the same category.

In clustering initially a word vector is created for the whole document which needs to be clustered.vector is numerical representation of vocab component-wise, value represents significance of word in documents. Then the distance matrix is used as input to an algorithm which outputs a group of clusters as per similarity.

2.2.4 Sort and compare sentences

We sorted the sentence vectors with respect to the distance from the centroid within the clusters. For sorting we have used python library which is adaptive merge sort algorithm having best time complexity as O(n) and worst case time complexity is O(n log n), Where n is the number of input. Now the sentences which have the same distance from centroid, we did string comparison within those sentences only (not with all the sentences). This is how we optimized the number of string comparisons which helps to reduce the overall time to find similar sentences among the documents.

3. EXPERIMENTS

Here we try to identify the minimum number of documents from which our proposed method beats brute force method (string comparison) in terms of overall completion time. We make use of 'TIME GRAPH' and experimented with various embedders along with various statistical methods.

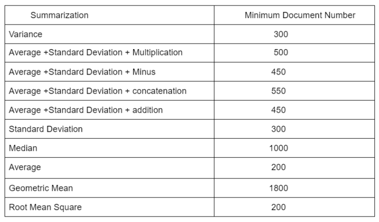
3.1 Time Graph

We plot graphs to do comparison study between brute force and our proposed method. Where x-axis is the total number of documents and y-axis is overall time taken to run the respective algorithm. Basically here we tried to identify the first crossing point in the graph which is the minimum number of documents where our algorithm started taking less time to complete the common text detection process than brute force method. [Refer figure 6 to 25 for this section]

3.2 Table

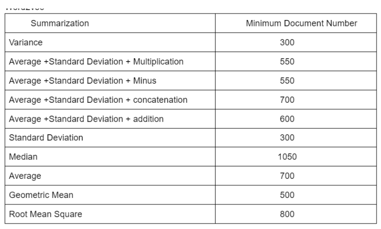
This is a table, where rows indicate the embedder used and columns indicate the statistics which we used to build sentences. In each cell we store the x-axis value of the crossing point (which is basically the minimum number of documents where our proposed algorithm started showing improvement over brute force) between two plots, [Refer table figure 3-Table-glove and figure-4 Table-word2vec for this experiment outcome individually]

In table figure-5 it shows the comparision between



Summarization	Minimum Document Number
Variance	300
Average +Standard Deviation + Multiplication	500
Average +Standard Deviation + Minus	450
Average +Standard Deviation + concatenation	550
Average +Standard Deviation + addition	450
Standard Deviation	300
Median	1000
Average	200
Geometric Mean	1800
Root Mean Square	200

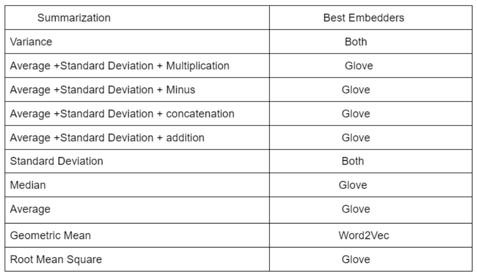
Figure 3: Experimented table with various feature extractions with Glove Embedders



Summarization	Minimum Document Number
Variance	300
Average +Standard Deviation + Multiplication	550
Average +Standard Deviation + Minus	550
Average +Standard Deviation + concatenation	700
Average +Standard Deviation + addition	600
Standard Deviation	300
Median	1050
Average	700
Geometric Mean	500
Root Mean Square	800

Figure 4: Experimented table with various feature extractions with Word2Vec Embedders

both the embedders and specify which is best among all the summarization. As we can observe that glove is working efficient in most of the summatizations.



Summarization	Best Embedders
Variance	Both
Average +Standard Deviation + Multiplication	Glove
Average +Standard Deviation + Minus	Glove
Average +Standard Deviation + concatenation	Glove
Average +Standard Deviation + addition	Glove
Standard Deviation	Both
Median	Glove
Average	Glove
Geometric Mean	Word2Vec
Root Mean Square	Glove

Figure 5: Experimented table with various feature extractions in comparision of best embedders among all the summarization [Note:- Here both means Glove and Word2Vec]

4. CONCLUSION

To detect similar sentences within given documents, in a normal string comparison method we have to compare all sentences to each document to each sentence of other documents, which is quite time consuming. To optimize string comparison between sentences, we took help of sentence vectors and clustering. To effectively build sentence vectors we have used different word embedders(glove and word to vec) and statics (mean, average etc) and observed which embedder statistic combination started showing improvement (in respect of

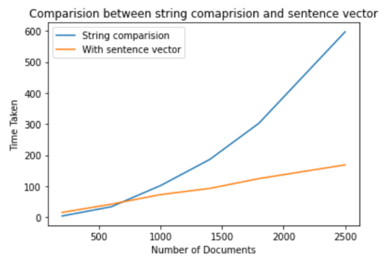


Figure 6: word2vec with average and standard deviation with concatenation

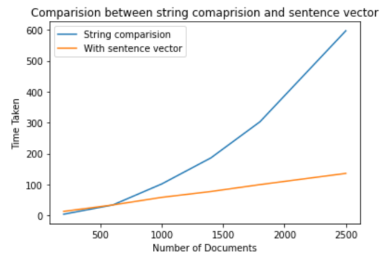


Figure 7: word2vec with average and standard deviation with minus

completion time) over brute force method at minimum number of documents. We observed that Glove word embedding with almost summarization (as static) performed best.
So, by this paper, we tried to convey that we can reduce the number of string comparisons between sentences by introducing the concept of sentence vector and clustering, which actually helps to complete the common text detection task in less time when there is a huge number of documents.

5. FUTURE WORK

Till now our algorithm can only detect exact similar sentences but if there is a case of synonym sentence it needs more approaches to find similarity.

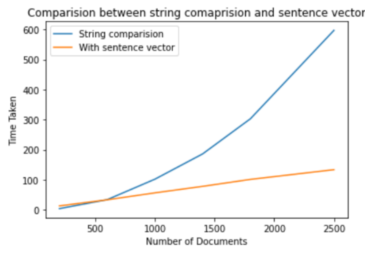


Figure 8: word2vec with average and standard deviation with multiplication

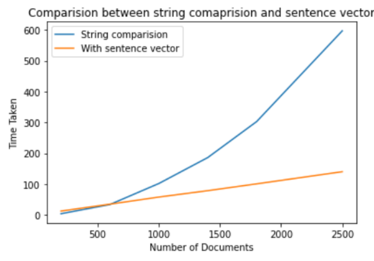


Figure 9: word2vec with average and standard deviation with addition

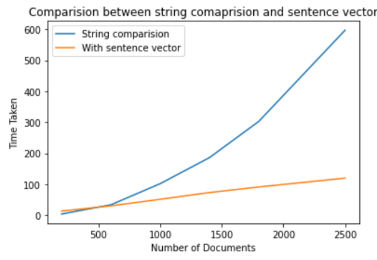


Figure 10: word2vec with geometrical mean

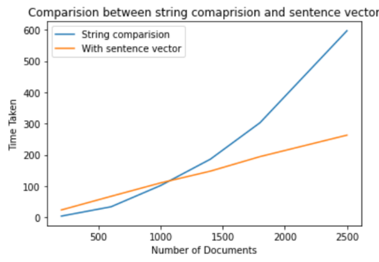


Figure 11: word2vec with median

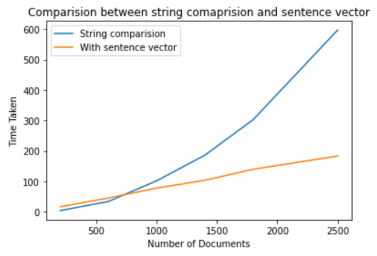


Figure 12: word2vec with average

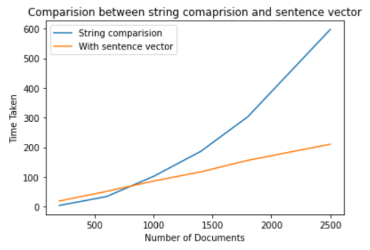


Figure 13: word2vec with root mean square

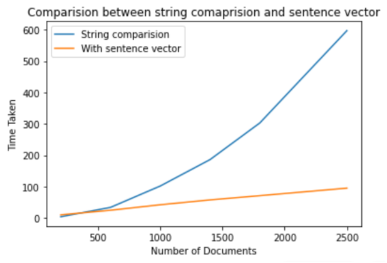


Figure 14: word2vec with standard deviation

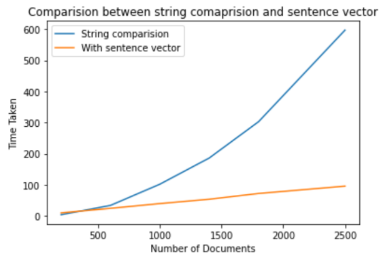


Figure 15: word2vec with variance

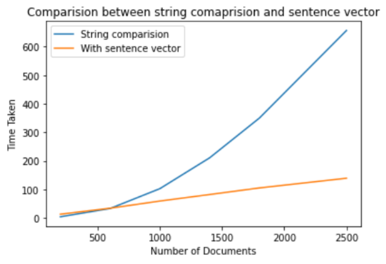


Figure 16: glove with average and standard deviation with concatenation

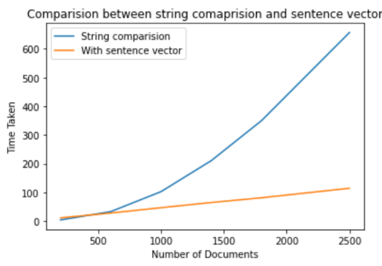


Figure 17: glove with average and standard deviation with minus

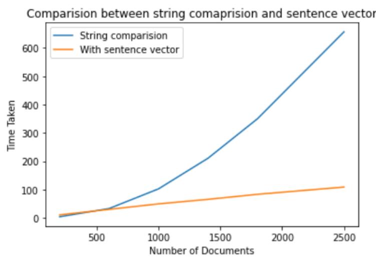


Figure 18: glove with average and standard deviation with multiplication

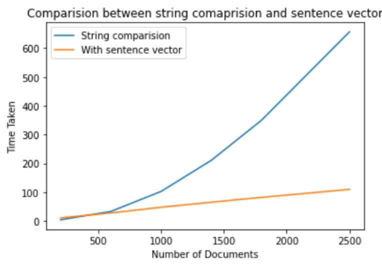


Figure 19: glove with average and standard deviation with addition

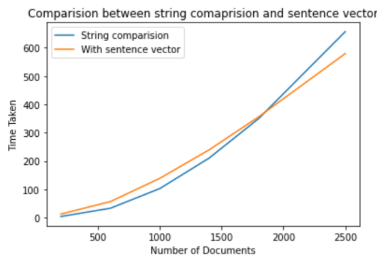


Figure 20: glove with geometrical mean

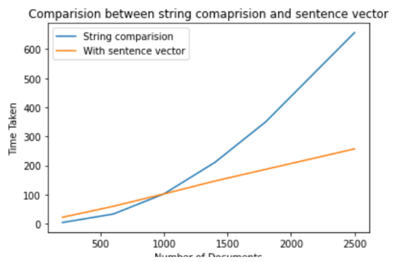


Figure 21: glove with median

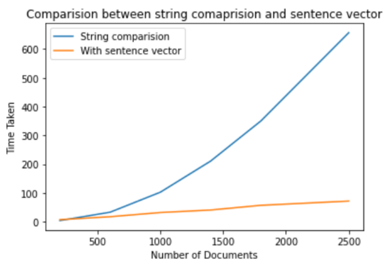


Figure 22: glove with average

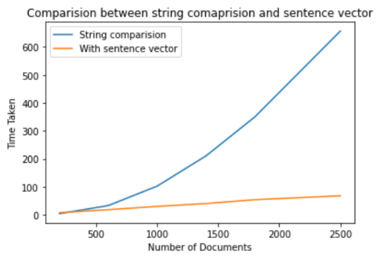


Figure 23: glove with root mean square

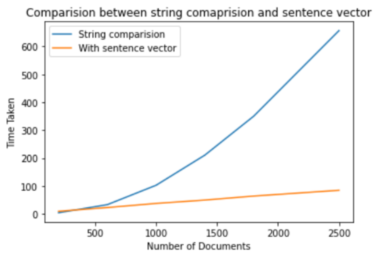


Figure 24: glove with standard deviation

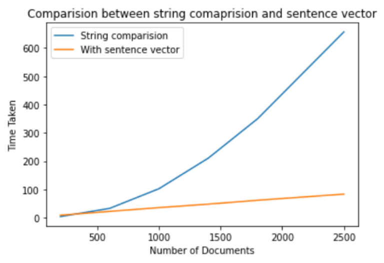


Figure 25: glove with variance

6. REFERENCES

[1]Journal of Intelligent Systems | Volume 25: Issue 3 Plagiarism Detection Using Machine Learning-Based Paraphrase Recognizer A. Chitra and Anupriya Rajkumar.

[2]S. Alzahrani, N. Salim and A. Abraham, Understanding plagiarism linguistic patterns, textual features and detection methods, IEEE T. Syst. Man Cyb.42 (2011), 133–149.

[3] Detecting short passage of similarity in large document collection,
http://www.cs.cornell.edu/home/llee/emnlp/papers/lyon.pdf

[4]On the resemblance and containment of documents Andrei Z. Broder,digital Systems Research Center 130 Lytton Avenue, Palo Alto, CA 94301, USA

[5]College of Saint Benedict and Saint John's University DigitalCommons@CSB/SJU Honors Theses, 1963-2015 Honors Program 2013 Efficient Clustering-based Plagiarism Detection using IPPDC Anthony Ohmann College of Saint Benedict/Saint John's University

[6]Waqar Ali, Tanveer Ahmad, Zobia Rehman, Anwar Ur Rehman, Munam Ali Shah, Ansar Abbas, Ghulam Dustgeer, "A Novel Framework for Plagiarism Detection: A Case Study for Urdu Language", Automation and Computing (ICAC) 2018 24th International Conference on, pp. 1-6, 2018.

[7]International Journal of Computer (IJC) ISSN 2307-4531,Thinn Lai Soe a University of Technology (Yatanarpon Cyber City) a thinnlaisoe@gmail.com.

[8]Andrey Kutuzov, Elizaveta Kuzmenko University of Oslo, National Research University Higher School of Economics andreku@ifi.uio.no, eakuzmenko 2@edu.hse.ru

[9]K. Vani and D. Gupta, "Using K-means cluster based techniques in external plagiarism detection," 2014 International Conference on Contemporary Computing and Informatics (IC3I),2014, pp. 1268-1273,
doi: 10.1109/IC3I.2014.7019659.

[10]Plagiarism Detection Process using Data Mining Techniques https://doi.org/10.3991/ijes.v5i4.7869 Mahwish Abid!!", Muhammad Usman, Muhammad Waleed Ashraf Riphah International University Faisalabad, Pak