*Text Summarization Using TextRank Algorithm*

NIKITA VISPUTE  
*Erik Johnsson School of Engineering*

*University of Texas at Dallas*Richardson, TX, USA

[nikita.vispute@utdallas.edu](mailto:nikita.vispute@utdallas.edu)

DIKSHA CHHABRA   
*Erik Johnsson School of Engineering*

*University of Texas at Dallas*Richardson, TX, USA

[dxc180003@utdallas.edu](mailto:dxc180003@utdallas.edu)

*Abstract*—With the increasing volume and variety in big data generated through various sources, it is becoming increasingly difficult to not only use this data efficiently but also to manually summarize large amounts of data into concise and useful information. This is one of the main reasons that text summarization is becoming a very popular research field in the Natural Language Processing (NLP). Text summarization is a process of picking a subset of sentences from the original text such that the selected sentences are a close representation of the original text in meaning. It is not about picking the most frequent or common words or entities but more about picking such sentences that preserve the semantic meaning of the information in the original text. There are various text summarization techniques that have been researched upon but almost all of them fall into two most fundamental approaches- extractive and abstractive. In this paper, we briefly describe the two main approaches of text summarization, then move on to describing the text summarization by text rank algorithm that we have implemented and lastly discuss the results and perform an analysis on the results achieved.

Keywords—text summarization, extractive method, text rank algorithm, tf-idf, rogue, summary.

# Introduction And Background Work

A summary is a concise form of the original text wherein the meaning of the original text is preserved with fewer sentences. This not only decreases the time required to read the large texts but is also requires less storage space.[3] Text summarization is the process of identifying the important features/sentences in the large text and compressing the original document to a brief concise description.[2] Text summarization is an important field in NLP. A document contains words, phrases, sentences, paragraphs, punctuation etc. The task of text summarization is not to find the most commonly occurring words but to extract those sentences that aptly summarize the document.[3]

Let us describe the various techniques of text summarization.

## Output Based Summarization [1]:

There are two main types: abstractive and extractive methods.

### Extractive Methods of summarization:

This method consists of selecting those a number of sentences that best represent the text document. The main sentences, paragraphs are selected and joined to form one summary that describes the document. The sentences are selected using some statistical or linguistic approaches (NLP methods). This is the most popular methods and can be implemented more easily than others.

### Abstractive Methods of summarization:

This method involves understanding the concept of the document first during the comprehension phase and then making a summary in a natural language by interpreting from the main essence of the document during the production phase. This does not necessarily use the original sentences and can change them or generate new sentences as well which are shorter representations of the text. This is a very difficult method as it is part of deep learning and is still being researched.

## Detail Based Summarization [1]:

There are two main approaches: informative and indicative methods.

### Informative summarization:

This summarization involves 20-30% of the original information and consists of the main points of the text.

### Indicative Summarization:

This method involves summarizing the original content such that it represents only 5-10% of the original data. This summarization is generally used to increase the curiosity of readers and have them read the full text. It is often applied in making prefaces, trailers etc.

## Content-based Summarization [1]:

There are two main types: Generic and Query-based.

### Generic summarization:

This method assumes that the reader of the summary has no understanding of the subject and may not have read the original text. This type of summary aims to give a complete all-round understanding of the topic and not just the salient features.

### Query-based summarization:

This type of summary assumes that the reader has already read the original text and has knowledge of the topic. It is usually aimed at answering specific questions that the reader may have

within a topic.

## Limitation-based Summarization [1]:

There are three types of this summarization approach: Independent, Domain-dependent, Genre-specific.

### Independent summarization:

This is much like the generic summary, where all the facts are considered in making the summary and is a general overview independent of the topic scope.

### Domain-dependent summarization:

In this type of summarization, specific input text of particular fields are accepted, and the summaries are generated based on the same. Fields can be like news, sports, tech etc.

### Genre-specific summarization:

This type of summarization is more specific than domain-dependent summarization. It involves focusing on a genre within a domain. Example summarization on pop culture genre from music domain.

## Number of input text-based Summarization [1]:

Based on the number of input text summarization can be of two types: single document, multi-document.

### Single Document Summarization:

In single document summarization, only one text document is used as an input and a summary is generated.

### Multi-Document Summarization:

Multi-document summarization is very popular these days and is an extension of single document summarization where the input for summarization process is a collection of related documents. The main aim of multi-document summarization is to get rid of redundancy across multiple documents and highlight the similarities and dissimilarities among them. There are two main approaches of doing this multi-document summarization process. The first one being the process of single document summarization where each document is summarized separately and then combined into one final summary after removing the redundancies in each through overlapping and comparison. The second approach is specifically designed for multi-document summarization where all the input documents are considered as one and one summary is generated by extracting important sentences from it using graph or clustering algorithms.

## Language acceptance-based Summarization [1]:

Language based summarization is classified into mono-language summarization, multi-language summarization.

### Mono-language Summarization:

This type of summarization accepts documents only in one language for the process.

### Multi-language Summarization:

This type of summarization accepts documents in many languages for the process and makes a summary considering contents of all them.

# Theoretical Study & Implementation Of The Algorithm

We have applied the TextRank algorithm to summarize the documents in our dataset. To do this we followed the following steps:

1. *Tokenization:*

Split the text into sentences and then each sentence into a set of words using the bag of words method.

1. *Similarity calculation:*

Calculated the similarity matrix for similarity among sentences.

1. *Build sentence graph:*

Built a graph of  the sentences

1. *TextRank:*

Scored the sentences using the PageRank Algorithm

1. *Summarization:*

Created the summary of the sentences based on the scores.

1. *Evaluation:*

Evaluated our results with the summaries provided in the data set in terms of precision, recall and F-measure.

Dataset used:

We used the BBC News summary dataset from Kaggle. The dataset contains two folders - ‘news articles’ and ‘summaries’ having  the articles from categories like tech, sports, business, entertainment and politics. The news articles contains the raw articles(.txt files) and their corresponding summary is given in summaries folder. We have used the summaries to measure the performance of algorithm, compare our results and evaluate it.

Preprocessing :

1. Tokenization:

We split each of the input text documents into sentences. For this, we used NLTK library and the Punkt module from this library. This step is called sentence identification. Next, we split the sentences into words using the bag of words model. In this model the sentences are split into an unordered collection of words and its occurence. The occurrence is usually given as a binary no: 1 for presence of the word in the sentence and 0 for absence in the sentence.

1. Count Vectorization:

After tokenization of the text into words we next apply the CountVectorizer. This helps to convert the collection of words from the input text documents into vectors of token counts.

1. Sentence Graph [4]

We got the output as a matrix where the rows are sentences and the columns are words from the previous step. Next, we transformed that into a graph where the sentences are related to each other. We first normalized our matrix using the Scikit-learn's TfidfTransformer method.  This creates the similarity matrix using the tf-idf (Term Frequency Inverse Document Frequency) technique. This is a mirrored matrix, where both the rows and columns correspond to sentences, and the elements describe how similar the two sentences are. Scores of 1 mean that the sentences are exactly the same, while scores of 0 mean the sentences have no overlap.

PageRank Algorithm [4]:

PageRank Algorithm developed by Larry Page is used for the ranking of web documents by the Google Internet search engine. The PageRank algorithm basically assigns weights to each element of a hyperlinked set of documents to measure its relative importance within the set. The algorithm can be applied to any collection of documents and of varying sizes. PageRank  of a page signifies the importance of the page. A hyperlink to a page counts as a vote of support. The PageRank of a page is defined recursively and depends on the number and PageRank metric of all pages that link to it ("incoming links"). A page that is linked to by many pages with high PageRank receives a high rank itself. If there are no links to a web page, then that page is relatively less important. The PageRank computations require several iterations, through the collection to give the most accurate value.

TextRank Algorithm [4]:

TextRank Algorithm is similar to PageRank, but PageRank is applied to web pages while TextRank is applied to texts for ranking purposes.

To apply the TextRank algorithm, we require our text in the form of a graph and the graph should depict relationships among the sentences. Vertices in the graph  can be words, entire sentences, etc. The application that determines the type of relations that are required to be draw connections between any two such vertices, e.g. lexical or semantic relations, contextual overlap, etc.  The goal of this implementation is to rank entire sentences, for appropriate text summarization and therefore a vertex is added to the graph for each sentence in the text. The resulting graph is highly connected, with a weight associated with each edge, indicating the strength of the connections established between various sentence pairs in the text. The text is therefore represented as a weighted graph. After the ranking algorithm is run on the graph, sentences are sorted in reversed order of their score. The sentences with the highest score are added to make the summary.

The following steps are applied:

1. Identify text units that best define the task and add them as vertices in the graph.
2. Identify relations that connect such text units and use these relations to draw edges between vertices in the graph. Edges can be directed or undirected, weighted or unweighted.
3. Iterate the graph-based ranking algorithm until convergence.
4. Sort vertices based on their final score. Use the values attached to each vertex for ranking/selection decisions.

In our implementation of the algorithm we used the graph of sentences we obtained from the tf-idf matrix and applied pagerank to score them. To do this, we used the pagerank function from NetworkX library from PySpark. Using this technique, we have a simple way of choosing relevant sentences from a text.

Evaluation:

We have used the ROUGE library in PySpark for evaluation of the performance of our algorithm. ROUGE stands for Recall-Oriented Understudy for Gisting Evaluation. It is  used for evaluation text summarization results by comparing the original text with the algorithm generated text summary.

We used 3 types of ROGUE techniques to evaluate summary:

1. ROGUE-N :

It stands for n-gram comparison of the sentence. We used 1-gram and 2-gram which is ROGUE-1 and ROGUE-2 in our results. ROGUE-1 uses the  number of words overlapping in both the summaries. ROGUE-2 is the number of bigrams overlapping in both the summaries.

1. ROGUE-L:

It stands for LCS (Longest Common Subsequence of the summaries). This doesn’t consider consecutive words rather it takes the words in a sequence occurring in the both the texts.

3. ROGUE-W:

It stands for weighted Longest Common Subsequence (LCS). It checks for consecutive LCSes.

Programming Language and Libraries used:

We have implemented our algorithm using PySpark in a databricks notebook and installed the following libraries as discussed earlier from PyPi:

* py-rouge===1.1
* nltk===3.4.5
* networkx===2.4
* s3fs

The input and output text files were retrieved from and stored in a bucket on AWS S3.

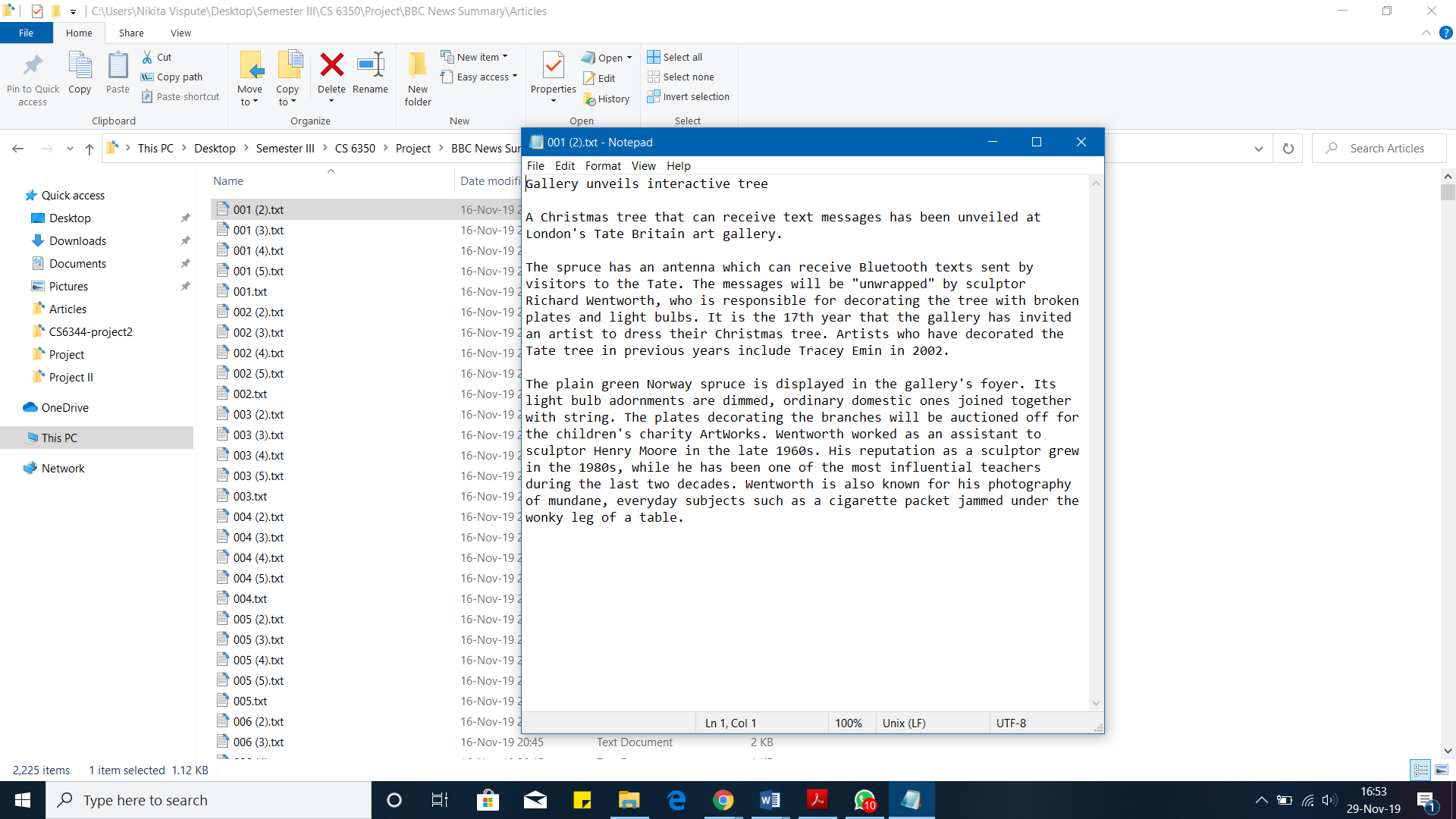
# Results And Analysis

After performing the above procedure, we have the following:

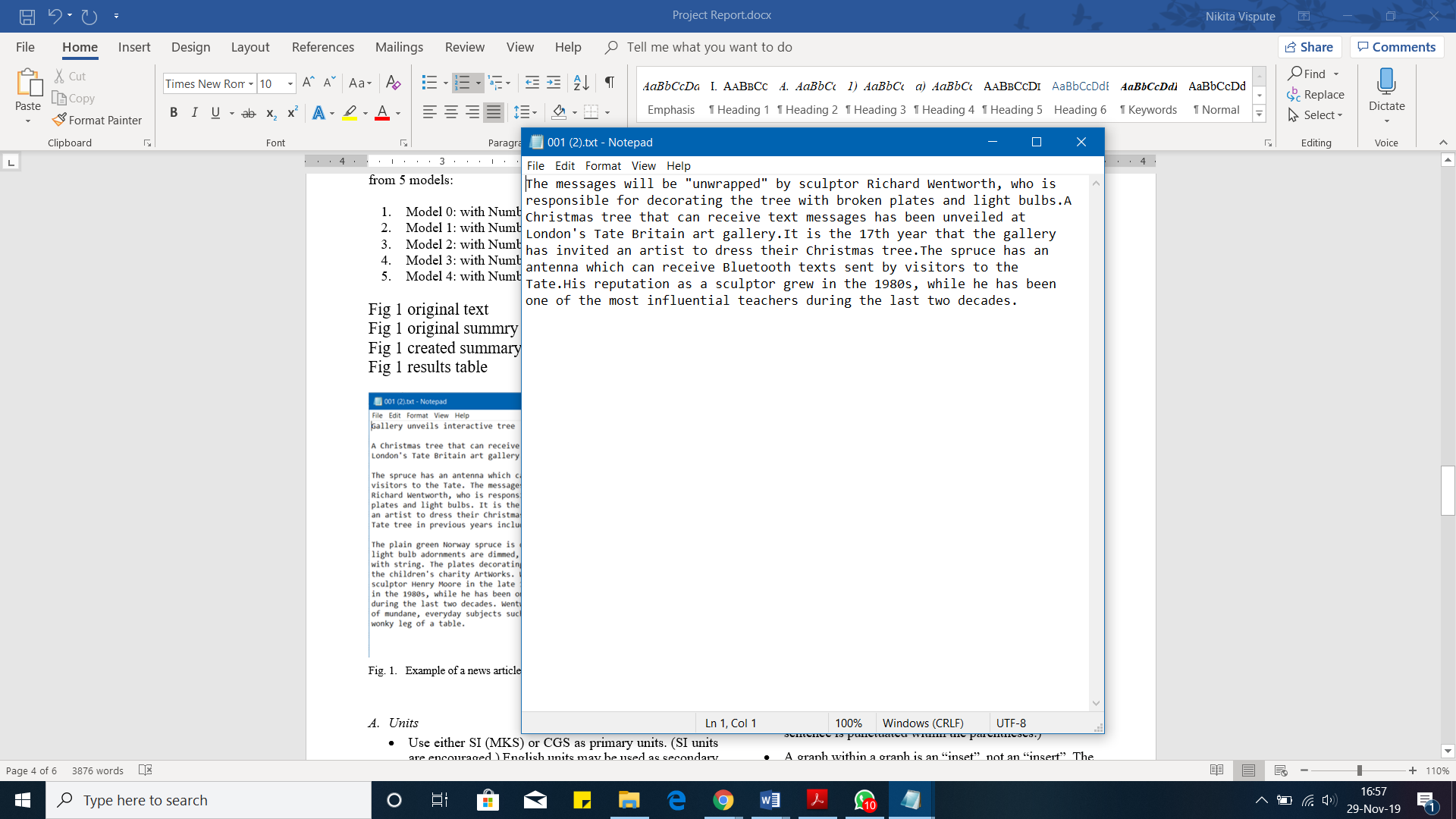
1. Original news articles “Fig. 1” and their summaries “Fig. 2” from the Kaggle webpage.
2. Algorithm generated summary of the news articles.
3. Results of the ROGUE[6] evaluation method.

We used Precision, Recall and F1-Score to measure the performance of our model. The following are the result we got from 5 models:

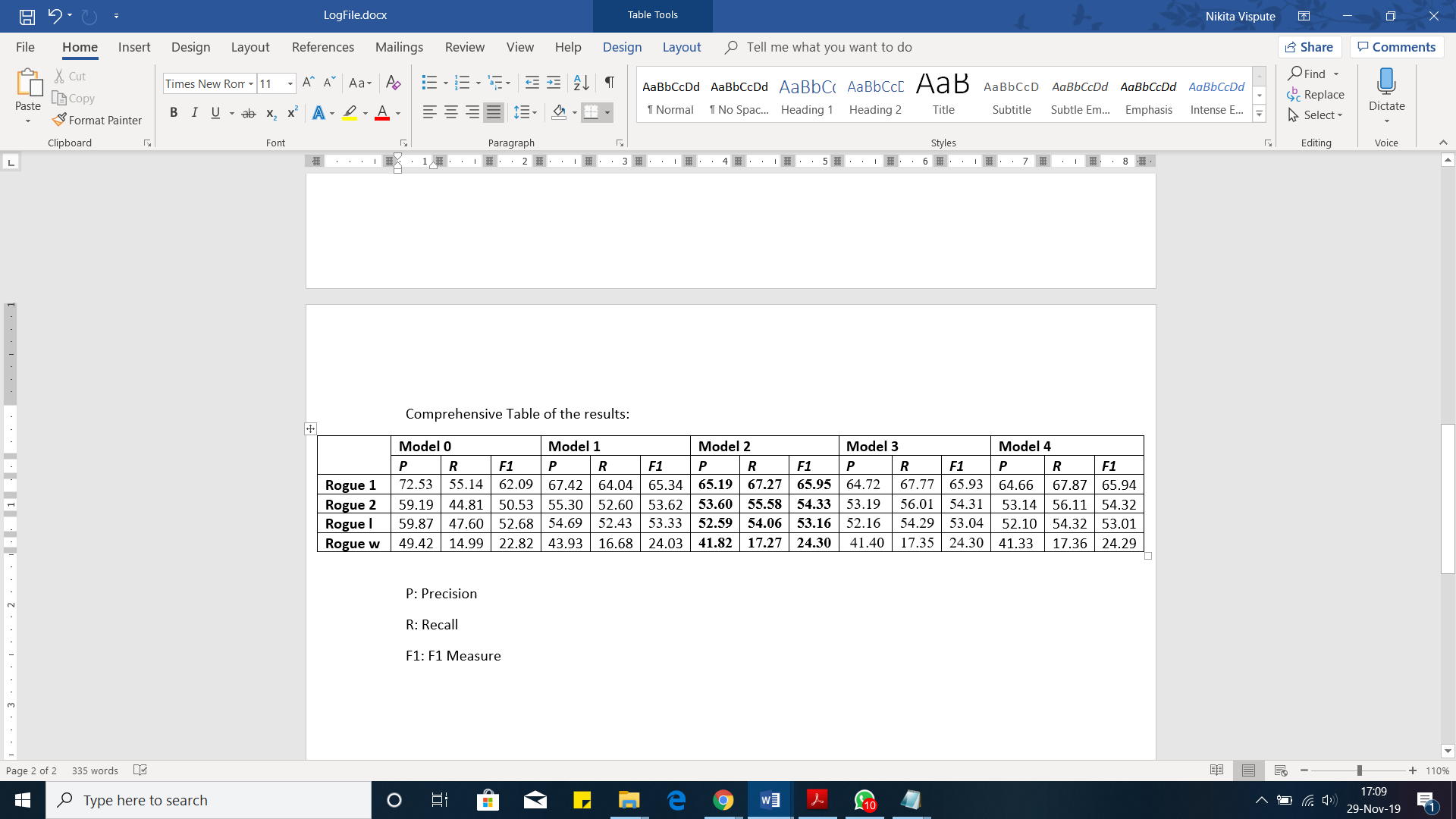
1. Model 0: with Number of sentences in the summary = 3
2. Model 1: with Number of sentences in the summary = 4
3. Model 2: with Number of sentences in the summary = 5
4. Model 3: with Number of sentences in the summary = 6
5. Model 4: with Number of sentences in the summary = 7

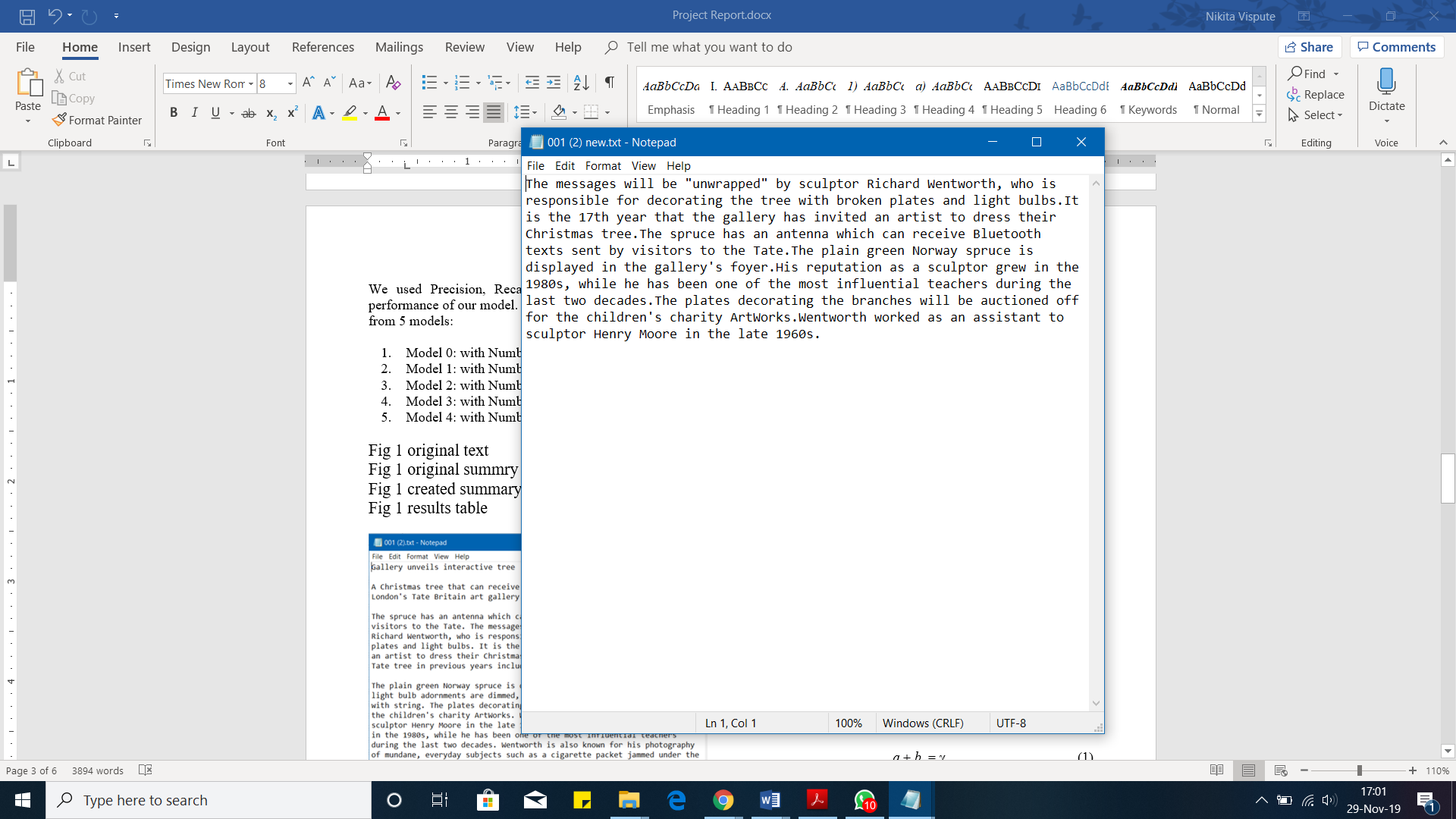


1. Example of a news article from the dataset.

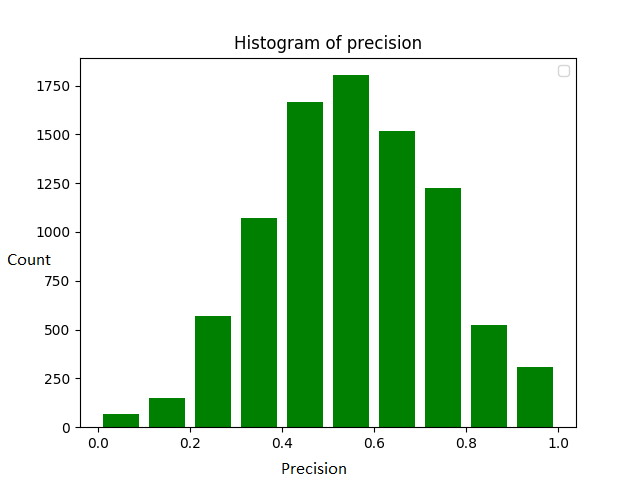


1. Example of a summary of the news article in Fig. 1. from the dataset

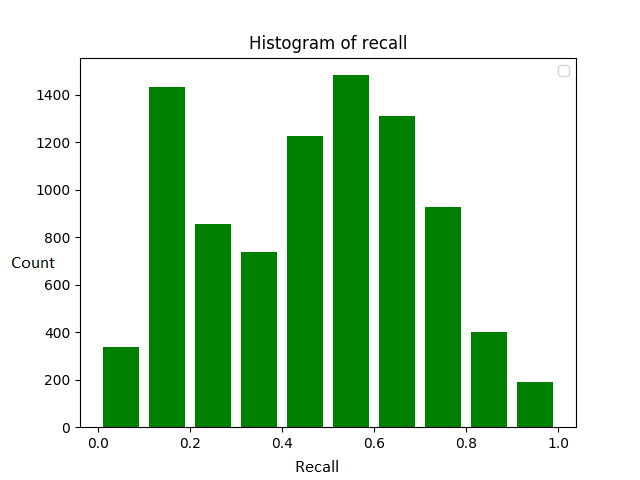
TABLE 1. RESULTS SHOWING PERFORMANCE OF OUR ALGORITHM IN TERMS OF PRECISION, RECALL, F1 MEASURE FOR ALL MODELS USING ROGUE 1, 2, L, W



1. Summary created by Model 4 with number of sentences in the summary as 7.



1. Histogram showing the precision against the count



1. Histogram showing the recall against the count

Observation and analysis:

1. We see that there is a trade-off between precision and recall in the models.
2. As the number of sentences in the summary increases, the Recall increases because there are more chances that the summary would contain similar words(unigrams, bigrams, LCS etc.).

Since Recall = number of matching words in algorithm-generated summary / number of words in reference summary,  the numerator increases but the denominator remains the same in this case.

1. As the number of sentences in the summary increases, Precision decreases because adding more summary might not result in any matching pairs but it would definitely increase the size of algorithm generated summary.

Since Precision = number of matching words in algorithm-generated summary / number of words in algorithm-generated summary, so the denominator increases more than the numerator and hence the overall precision decreases.

1. Bigram results (rouge-2) are lower than Unigram(rogue-1) since occurring of two consecutive words in both the reference and algorithm generated summary is very less.
2. On the other hand, Unigram results are very high as it is matching single words in both the reference and algorithm generated summary.
3. F1 score is the weighted average of Precision and Recall and so it would consider both of them for model evaluation. This score increases till Model 2 and then decreases. This would be a better measure of the model evaluation. We can see that Model 2 has the highest F1 score (65.95) in rogue-1, which considers the 5 sentences of text summary. (“Table. 1”)

Hence, by looking at the F1 scores we can say that Model 2 with summary sentence size=5 is the best model in our case.

We observe from the graphs (“Fig. 4”, “Fig. 5”) that there are fewer high values present in the recall curve than the precision curve. Precision curve follows a Gaussian distribution where most of the values are centered around 0.5 while in recall curve, there are also a high number of points with less than 0.2 recall value. On the whole, their weighted average which is the F1 score proves out to be the best among all the models.

# Conclusion and future work

Future work involves modifying the algorithm to incorporate sentence and word embedding using the word2vec[7] models for deep-learning-based text summarization. A mixture of CNNs with RNNs might be useful for getting better precision and recall values. Also, a technique called skip-gram which takes the words in order (not necessarily consecutive) for comparing reference and algorithm generated text might help in the evaluation of the models.

Though we implemented a naive method of text summarization, but there are other techniques available to better implement the NN based techniques which require better hardware and computing power. In our case we took the 5 models with different number of sentences, performed the text summarization algorithm on them and calculated the scores using the ROGUE technique. We took the best model as the one having 5 number of sentences with the highest F1-Score. We can conclude that it is not necessary that including more and more sentences in the summary would improve the results as in our case. Getting a good number as the number which maximizes the score of the summary details is important to achieve a balance of precision and recall and hence produces an informative and well-formed summary.

##### References

1. S. R. Rahimi, A. T. Mozhdehi, and M. Abdolahi, “An overview on extractive text summarization,” 2017 IEEE 4th International Conference on KBEI, December 2017.
2. J. N. Madhuri, and G. Kumar, “Extractive text summarization using sentence ranking,” 2019.
3. S. Biswas, R. Rautray, R. Dash, and R. Dash, “Text summarization: a review,” 2018 2nd International Conference on Data Science and Business Analytics, 2018.
4. K. S. Thakkar, Dr. R. V. Dharaskar, and M. B. Chandak, “Graph-Based Algorithms for Text Summarization,” Third International Conference on Emerging Trends in Engineering and Technology, 2010.
5. L. Page, et al. “PageRank: Bringing order to the web,” Stanford Digital Libraries Working Paper, vol. 72, 1997.
6. C. Y. Lin, “Rouge: A package for automatic evaluation of summaries,” Text summarization branches out, 2004.
7. Goldberg, Yoav, and O. Levy, “word2vec explained: deriving Mikolov et al.'s negative-sampling word-embedding method,” arXiv preprint arXiv:1402.3722, 2014.
8. <https://joshbohde.com/blog/document-summarization/>