Graph based Methods for Text Summarization

Darragh Clabby
HDAIML_SEP
20142935
National College of Ireland
Dublin, Ireland
x20142935@student.ncirl.ie

***Abstract*—This project explores the application of the TextRank algorithm to the task of automatic text summarization. The TextRank algorithm ranks the importance of sentences in a text by representing the text as a graph, with vertices representing sentences and edges representing similarity between sentences. The texts to be summarized were news articles taken from the Daily Mail data set. Each text contained in the data set is accompanied by a human generated summary which may be used as reference summary against which automatically generated summaries may be assessed. Summarization was implemented according to five methods, three of which employed the TextRank algorithm. The aim of this work was to assess how the basic TextRank method compared to the simpler TF-IDF method; and to assess whether sentence embedding and unsupervised classification could be employed to further improve performance. The results suggest that the basic TextRank performed best (highest accuracy, lowest computational demand). The methods employing sentence embedding and unsupervised classification did not deliver further performance improvements.**

Keywords—Text summarization, TextRank algorithm

I. Introduction

In recent years an extremely large volume of data has become available via internet sources. This data may be expressed in terms of text, images, videos, etc. This often leads to "information overload" whereby the excessive quantity of data available means that individuals find it difficult to make sense of an issue, and struggle to make decisions on an issue. Summarization of available data may be employed to combat information overload.

In the case of textual data (e.g. news articles, scientific papers, etc.) summarization was defined by Radev [1] as "a text that is produced from one or more texts, that conveys important information in the original text(s), and that is no longer than half of the original text(s) and usually, significantly less than that".

Most people are familiar with the process of manual summarization, and in particular the significant level of effort required, particularly as the volume of data to be considered increases. As a result, automation of this process such that summarization texts may be generated by machines represents a significant area of research in the field of artificial intelligence and natural language processing.

A variety of approaches have been employed to achieve automatic text summarization. These approaches range in complexity from simple term frequency ranking algorithms to advanced methods such as Generative Adversarial Networks (GANs) [2]. Graph based methods such as the TextRank algorithm [3] represent an interesting approach in that they can achieve good performance despite the relative simplicity of the algorithm. The motivation for this work is therefore to explore graph based text summarization methods (in particular the TextRank algorithm) with the aim of addressing the following research questions:

- How does the performance of a graph based text summarization method (TextRank) compare to a simpler method (TF-IDF);
- Can improvements to the graph based method be identified?

II. Related Work

A. Related Methods

The main method relevant to this work is the TextRank algorithm presented by Mihalcea [4]. TextRank is a graph based method of ranking sentences. It is based on Google's PageRank algorithm which was developed to analyse links between web pages and rank them accordingly. The basic idea of TextRank is that documents are represented as a graph, where vertices represent sentences and edges represent similarity between sentences. The algorithm recursively ranks the importance of each vertex based on its similarity to other sentences, and its own rank.

Another key method relevant to this work is TF-IDF (Term Frequency Inverse Document Frequency), an overview of which is given by Allahyari [5]. This may be viewed as a more basic method of ranking sentences in a text based on the frequency with which words appear in the document. However, considering term frequency gives equal weight to content words and function words. In order to increase the importance of content words the IDF coefficient is employed. This coefficient reduces the weight of words as the number of documents in which they appear increases (assuming a corpus of more than one document). This means that stop words can be automatically identified and effectively filtered.

Finally, unsupervised classification of embedded sentence vectors has been employed as a method to identify clusters of sentences with similar semantic content. This approach has been used by Padmakumar [6] as a basis for implementing text summarization.

B. Related Works

The approach presented by Thakar [7] informed this work. Thakar presented text summarization of a single document using three methods: term frequency, TextRank; and unsupervised classification. While the methods used were considered interesting, the analysis was relatively limited (only a single document was considered) and superficial (results were assessed subjectively, no quantitative analysis was implemented).

Furthermore, Thakar drew on the works of other authors, including Chauhan [8] which formed the basis for the unsupervised classification method. The unsupervised classification method presented by Chuahan formed the basis for the ClusterCenter method employed in this work. However, the following differences between the present work and that presented by Chuahan should be noted:

- Chuahan trained a skip-thought encoder/decoder network based on data from Wikipedia to perform sentence embeddings;

- Chuahan's method was applied to emails, which are expected to be shorter than the news articles considered in the present work;
- Chuahan did not have reference summaries against which results could be assessed. As a result, analysis of results was limited to a purely qualitative assessment.

Given the limitations of these two related works, the work presented in this document seeks to apply similar methods (TF-IDF, TextRank, ClusterCenter) to a larger set of documents and implement a more rigorous quantitative assessment of the results. Furthermore, the present work seeks to explore additional refined/improved methods by employing sentence embeddings (CosRank) and unsupervised classification (ClusterRank) in conjunction with the TextRank algorithm.

III. Methodology

The data set used for this work was the Daily Mail data set, which in conjunction with the CNN data set is commonly used for development of automatic text summarization methods. Five separate methods were employed to generate summaries of news articles taken from this data set. Three of the five methods employed the TextRank algorithm to rank sentences, while the remaining two served as baselines against which aspects of the three graph based methods could be assessed. For each method tests were implemented across a range corpus size (10, 100, 1000 documents). Performance was assessed in terms of accuracy relative to each text's human generated reference summary, execution time, and length of the generated summary (in terms of number of words).

A. Data Preprocessing

Two data sets commonly used for text summarization are the CNN and Daily Mail data sets presented by Hermann [9]. These data sets comprise news articles accompanied by human generated summaries. They are therefore particularly well suited to evaluation of text summarization methods since machine generated summaries derived from the article's main text may be quantitatively assessed against the article's human generated summary.

For the purpose of this work the Daily Mail data set was used. The Stories data set provided by Cho [10] was downloaded. It comprises 219,507 news articles. Each file is structured such that the main text appears at the start, followed by a number of summary lines. The summary lines are delimited by the "@highlight" tag which appears at the start of each summary line.

1) Preprocessing of Raw Files

Preprocessing of a raw file was implemented according to the following steps:

- For a given file all text was concatenated into a single string;
- The full text string was split on the "@highlight" tag. The result was a list of strings of length n+1, where n was the number of times that the "@highlight" tag appeared in the file;
- The first string in the resulting list represents all text that appeared before the first "@highlight" tag. It therefore corresponds to the main text;
- All strings other than the first string of the list represent a highlight. Each highlight was concatenated into a single string to form the summary;

The procedure outlined above was implemented for a specified subset of the entire data set (e.g. 1000 of the 219,507 story files). For each file, the summary and main text were stored in a data frame along with the file name and the number of lines in the summary. This data frame represents the corpus.

2) Handling New Line Tags

Following preprocessing the text may retain new line tags such as "\n" and "\xa0". In order to remove these tags they were replaced by a space followed by a full stop (" ."). The reason that they were not simply deleted or replaced by a blank space is that they may or may not follow sentences that end with a full stop. Addition of " ." ensures that sentences not terminated with a full stop will be tokenized as intended. For sentences that were terminated by a full step, sentence tokenization will produce an additional string containing only a space. This is acceptable since blank strings may be easily discarded.

3) Sentence Tokenization

The next step was to split the text into a list of sentences, i.e. sentence tokenization. This was implemented using python's NLTK (Natural language Tool Kit) library, specifically the sent_tokenize method of the tokenize object. The result is a list of strings where each string is a separate sentence of the text.

4) Sentence Cleaning

For each sentence punctuation and digits were removed, and all characters were converted to lower case. Stop words (as specified in nltk's corpus object) could also be optionally removed.

5) Word Tokenization

Finally, each cleaned sentence was tokenized using the word_tokenize method of nltk's tokenize object. If this step produced an empty list (i.e. the sentence had no remaining characters after removal of punctuation, digits, stop words, and subsequent word tokenization) nothing was added to the output. Otherwise the following items were added to the output:

- the list of words produced by the word tokenization;
- the original sentence used to produce the list of words.

6) Result

The result of this process is two nested lists containing the clean words (cleanText) and the corresponding raw sentences (rawText). The cleanText list has three levels corresponding to the document, sentence, and word. The rawText list has two levels corresponding to the document and sentence. For example:

- cleanText[i][j][k] gives the kth word in the jth sentence of the ith document;
- rawText[i][j] gives the jth sentence of the ith document.

B. Text Summarization Methods

For this work, five separate text summarization methods were implemented. These methods were implemented using python functions specifically developed for this project. These functions used the following libraries:

- Utility libraries: pandas, numpy, string, os, time;
- Natural language processing: nltk;
- Sentence embedding: sentence_transformers.

The python functions developed for the text summarization methods implemented in this project are described in Table 1. Note that functions for each method are categorized in terms of the general task categories for text summarization presented by Allahyari [2], namely:

- Analysis: constructing a representation of the text that conveys important information;
- Transformation: score the sentences based on the representation;
- Synthesis: generate a summary based on the results of sentence scoring.

The following subsections present a brief description of each of the five methods considered in this work.

TABLE I. PYTHON FUNCTIONS USED TO IMPLEMENT TEXT SUMMARIZATION METHODS

Method	*Analysis*	*Transformation*	*Synthesis*
TF-IDF	calcTFIDFcoeffs	calcScoresTFIDF	generateSummary
TextRank	sentenceSimilarity	calcScoresTextRank	generateSummary
CosRank	generateSentenceVecs, sentenceEmbeddingSimilarity	calcScoresTextRank	generateSummary
ClusterCenter	generateSentenceVecs, knn	calcClusterCentres	generateClusterSummaries
ClusterRank	generateSentenceVecs, knn	calcClusterRank	generateClusterSummaries

1) TF-IDF

A commonly employed approach for analysis and ranking of sentences for text summarization is TF-IDF (Term Frequency Inverse Document Frequency). This approach was included as part of this work since it may be considered as one of the most basic approaches for text summarization. It was therefore intended to serve as a baseline against which the more advanced graph based approaches using TextRank could be assessed.

TF-IDF involves two parts. First, the term frequency (TF) is calculated over a single document. This is simply the ratio of a_{ij} (the number of times word j appears document i) to the total number of words in the document. The second step is the inverse document frequency (IDF) which is calculated over the entire corpus. This set of coefficients is the log of the total number of documents N divided by the number of documents in which word j appears ($\Sigma_i b_{ij}$, where b_{ij} is 1 if word j appears in document i, and 0 otherwise).

$$TFIDF_{ij} = \left(a_{ij} / \sum_j a_{ij}\right) log\left(N / \sum_i b_{ij}\right)$$

Calculation of coefficients was implemented in the calcTFIDFcoeffs function. The sentences were then scored using the calcScoresTFIDF. Scoring sentences in a given document was achieved by summing the relevant terms for each word in the sentence. In the implementation used in this work, sentence scores were divided by the number of words in the sentence to avoid bias towards longer sentences.

The sentence scores were then passed to the generateSummary function which extracted the sentences with the highest scores. In all cases, the top four sentences were considered, since the human generated summaries in the Daily Mail data set were on average four sentences long. The summary was then arranged in the order that they appeared in the original text.

2) TextRank

The basic TextRank method considered for this work adhered to the method described by [4] for sentence extraction. The purpose of including this method was: to assess the performance of a graph based approach relative to the simpler TF-IDF approach; and to serve as a baseline against which alternative approaches employing the TextRank algorithm could be assessed.

For this implementation sentence analysis was implemented in the sentenceSimilarity function. This involved calculation of s_{ij}, the similarity between sentences S_i and S_j, which is the number of words w_k that appear in both sentences divided by the sum of the logs of the number of words in each sentence:

$$s_{ij} = |\{w_k | w_k \in S_i \,\&\, w_k \in S_j\}| / \left(log(|S_i|) + log(|S_j|)\right)$$

Sentences were then scored according to the TextRank algorithm using the calcScoresTextRank function. The graph was constructed with vertex scores V_i and edge weights (e.g. between vertices i and j) denoted w_{ij}. Vertex scores were initialized to a value V = 1 and a damping factor d=0.85 was employed (based on the recommendation given by [4]). The algorithm then iterated until the error (defined as the total difference in vertex scores between successive iterations) was less than 0.0001.

$$V_i = (1 - d) + d \sum_j \frac{w_{ji}}{\sum_k w_{jk}} V_j$$

Summaries were generated using the generateSummary function in the same manner as described for the TF-IDF method.

3) CosRank

The CosRank method was included as it was considered as an advance on the basic TextRank method described in the previous section. Indeed, [4] mentions that cosine similarity would be considered as part of future work. Furthermore, since the present work employs a more recent sentence embedding model, SBERT sentence transformer [11], it may be considered as an advance on the original work.

For this method, analysis of sentences entails vector embedding. A common approach to sentence embedding is to use word embeddings for each word in the sentence and then take a weighted average over vector features to produce a single vector for the sentence. A more direct approach is to use sentence embedding. In this work a pre trained SBERT model was used [11]. The “stsb-roberta-base” model was used via the sentence-transformers python library [12]. This model was chosen since it is described as being optimized for semantic textual similarity, and this was considered appropriate for the task under consideration.

Once the model was loaded using the SentenceTransformer method of the sentence_transformers library, it was used as part of the generateSentenceVecs function to generate sentence vectors. The sentenceEmbeddingSimilarity function was then used to

calculate cosine similarity between sentence vectors u_i and u_j, i.e.

$$c_{ij} = \frac{u_i \cdot u_j}{|u_i||u_j|}$$

Sentences were then ranked using the calcScoresTextRank function implemented in the same manner as described in the previous section. Finally, summaries were generated using the generateSummary function, as described previously.

4) *ClusterCenter*

This method broadly followed that presented by [8]. It was included as a baseline for the application of unsupervised learning techniques to the task of text summarization.

In this method sentence embeddings were generated according to the same procedure as described in the previous section. The calcClusterCentres function was then used to cluster the vectors with a predefined number of clusters, k. The default value was k=4 since this corresponds to the number of sentences to be included in the summaries. Clustering was implemented by the knn function. Cluster centers were randomly initialized. However, the random number generator was seeded with a value of 0 to ensure repeatability.

Sentences in each cluster were then scored based on their proximity to the cluster center, where:

- The cluster center was taken as the average of the vectors contained in the cluster;
- Proximity of each sentence vector to the center was calculated based on cosine similarity. Note that euclidean distance was initially employed but it was found to give poor results as sentences appeared to be clustered based on their length rather than their semantic content.

Summaries were generated using the generateClusterSummaries summaries function. This was similar to the generateSummary function used for the three previously described methods. However, summaries were generated separately for each cluster. Furthermore, the number of sentences extracted from each cluster was weighted based on the number of elements contained in the cluster. This was implemented to prevent over-representation of clusters with a small number of vectors. This meant that summaries may omit sentences from small clusters, and may contain multiple sentences from large clusters.

5) *ClusterRank*

The final method to be considered was the ClusterRank method. This was considered to assess whether unsupervised classification of sentences could improve performance of the basic TextRank method. Stated alternatively, this method may be used to assess whether the TextRank algorithm represents an improvement to the unsupervised method described in the previous section.

The implementation of this method was the same as that described for the ClusterCenter method, with the exception of the sentence scoring step. Rather than scoring sentences in a given cluster based on their proximity to the center of the cluster, the TextRank algorithm was applied (separately) to sentences in each cluster. This was implemented using the calcClusterRank function. Once sentences were scored by applying the TextRank algorithm to each cluster, summaries were generated using the generateClusterSummaries function, as described in the previous section.

IV. Evaluation

C. *Evaluation Criteria*

The following evaluation criteria were used to quantitatively assess the performance of each method: accuracy; execution time; summary length.

1) *Accuracy*

For machine generated text summarization, the ROGUE metric (Recall-Oriented Understudy for Gisting Evaluation) presented by Lin [13] is commonly used to quantify accuracy. This metric quantifies a machine generated summary's accuracy relative to a human generated summary by counting the number of overlapping terms (e.g. words) between them. For this work the ROGUE-N statistic was used, with unigrams (N=1) considered. Calculation of the ROGUE score was implemented in the calcRogue function. For the case of unigram (N=1) and a single reference summary, the rogue score is given by the number of words that occur in both the candidate and reference summaries divided by the number of words in the reference summary, i.e.

$$ROGUE = |\{w_c\} \cap \{w_r\}|/|\{w_r\}|$$

where $\{w_c\}$ and $\{w_r\}$ are the sets of words in the candidate and reference summaries respectively.

2) *Execution Time*

The time for each method to generate summaries for all documents in the corpus was recorded. The execution times presented in this report should be considered in the context of the following machine specifications:

- CPU: Intel®Core™i7-4910MQ CPU@2.90GHz×8
- RAM: 24GB

3) *Summary Length*

For all methods considered in this work summaries were generated to have a length of four sentences. This was chosen since the average number of sentences in the reference summaries was four. In order to compare reference and candidate summaries in terms of word counts, the following metric was considered:

$$L = |\{w_r\}|/|\{w_c\}|$$

This ratio measures how concise a candidate summary is relative to the reference summary. Higher values (>1) indicate a candidate summary that is shorter than its reference summary, i.e. $|\{w_c\}| < |\{w_r\}|$.

D. *Results*

Results are considered in both qualitative and quantitative terms.

1) *Qualitative Assessment*

Appendix A presents the original text, reference summary, and the candidate summaries generated by each method for a single story in the data set. The following general observations are noted with regard to this example:

- All five candidate summaries contain redundancy as they contain at least one of the two pairs of duplicate sentences (highlighted yellow and cyan) that appear in the original text. In the case of the summary produced by the TextRank method, there are only two unique sentences, since it is comprised of both pairs of duplicate sentences;

- The TF-IDF method includes a sentence with little semantic content: “The groundhog day special cake” (highlighted purple);
- The methods employing unsupervised classification (ClusterCenter and ClusterRank) seem to exhibit a less fluent sentence sequence. This is likely due to the fact that summaries for clusters were generated separately and concatenated. As a result they do not necessarily adhere to the original sentence ordering.

2) Quantitative Assessment

This section presents a quantitative assessment of summarization methods assessed over a corpus of 1000 documents. Figure 1 shows the average accuracy of the summaries produced by each method. The following observations are noted:

- The TextRank method had the highest average accuracy;
- All three methods employing the TextRank algorithm (TextRank, CosRank, ClusterRank) had higher accuracy than those that did not (TF-IDF, ClusterCenter);
- Methods that were anticipated to represent an advance on the basic TextRank method (i.e. CosRank, ClusterRank) did not show improved accuracy;

Figure 2 shows the execution time for each method over the 1000 document corpus. The following observations are noted:

- The TF-IDF method has the highest execution time.
- The methods employing sentence embeddings had comparable execution times. This was likely due to the fact that their execution times were dominated by the time taken to generate embedded sentence vectors;
- The execution time for the basic TextRank method was an order of magnitude lower than those for all other methods, i.e. ~2 minutes compared to ~30 minutes.

Finally, Figure 3 shows the average word length ratio for each method. This implies that all methods other than TF-IDF generate summaries with a larger number of words relative to the reference summaries.

E. Error Analysis

1) Sensitivity to Size of Corpus

Figure 4 shows the sensitivity of accuracy to corpus size, when considered across 10, 100, and 1000 documents. This shows that improved accuracy associated with the three methods employing the TextRank algorithm relative to those that do not is consistent across the three corpus sizes considered.

Figure 5 shows the sensitivity of execution time to corpus size (with both axes on a logarithmic scale). This suggests that the TextRank method would scale better than all other methods. Furthermore, the trend shown for the TF-IDF method implies that it would scale worst of all methods considered.

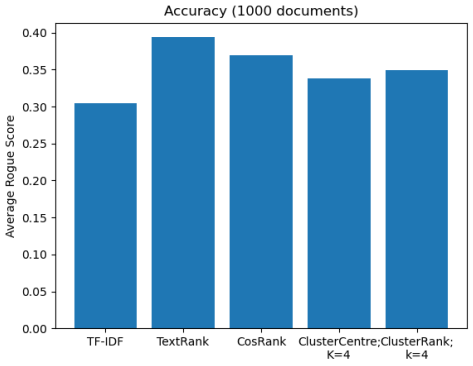

Fig. 1. Accuracy of each method assessed based on average Rogue scores over corpus of 1000 documents

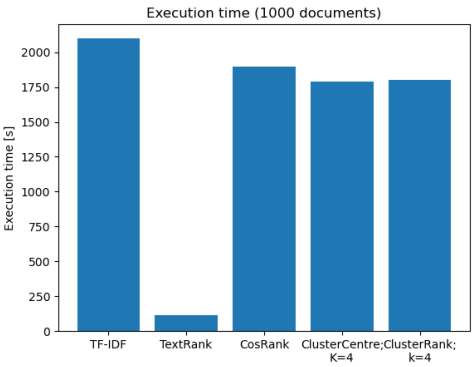

Fig. 2. Execution time for each method assessed based on a corpus of 1000 documents

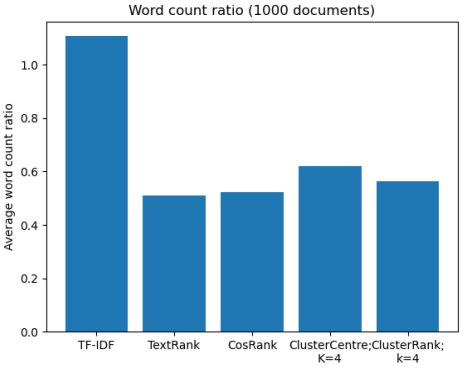

Fig. 3. Average word count ratio for each method based on a corpus of 1000 documents

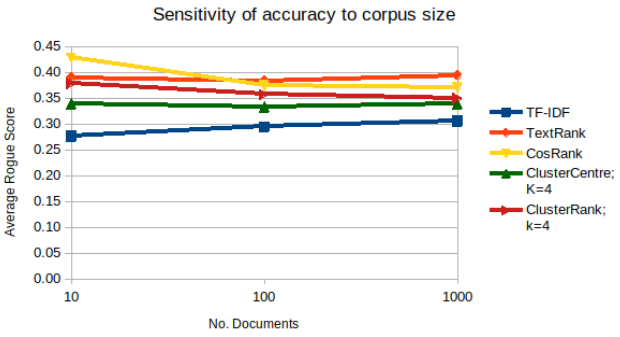

Fig. 4. Sensitivity of accuracy to corpus size

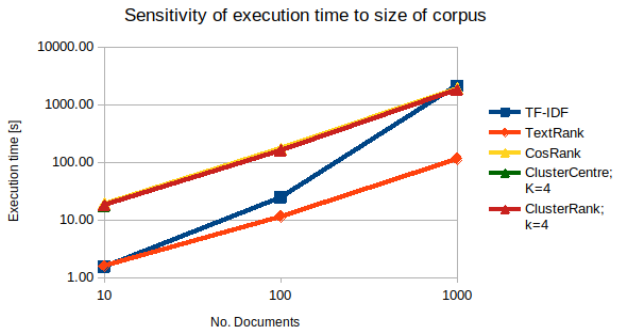

Fig. 5. Sensitivity of execution time to corpus size

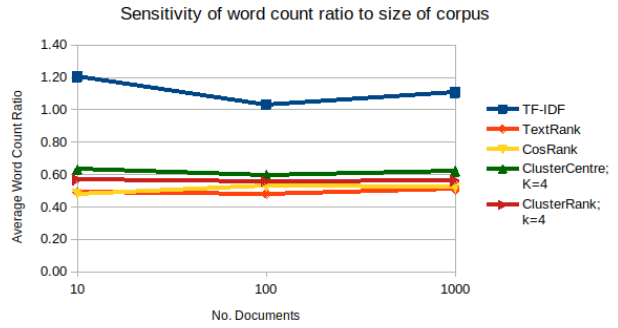

Fig. 6. Sensitivity of word count ratio to corpus size

Figure 6 shows the sensitivity of word count ratio to corpus size. This shows that the tendency of all methods other than TF-IDF to generate summaries with more words relative to the summary is consistent across the range of corpus size considered.

2) Sensitivity of Unsupervised Classification Methods to Number of Clusters

For the two unsupervised classification methods (ClusterCenter and ClusterRank) the default number of clusters was set to k=4 (based on the number of sentences that the summaries would contain). The sensitivity of summary accuracy to k was assessed for a 100 document corpus size. The solid lines in Figure 7 show the average accuracy for k in the range 1 to 4. Also shown in Figure 7 by the yellow dashed line is the average accuracy for the CosRank method. This was included as a reference point since it is equivalent to ClusterRank with k=1.

Also shown in Figure 7 are the maximum accuracies for each method. For a given document the maximum accuracy over k=1, 2, 3, 4 was identified. The average accuracy over all documents was then calculated considering the value of k that gave maximum accuracy for each document. This is shown by the red and green dashed lines in Figure 7. These statistics may be viewed as an estimate of the upper bound on the accuracy that could be achieved if k was optimized for each document. The results shown in Figure 7 suggest that average accuracies exceeding 0.4 would be possible with optimization of k. This would represent an improvement on all other methods considered. However, it requires automatic identification of the optimal number of clusters for each document.

3) Variability of Accuracy

Figure 8 shows a box and whisker plot for the accuracy of each method assessed over a 1000 document corpus. This is included to illustrate that all methods exhibit significant variability in accuracy.

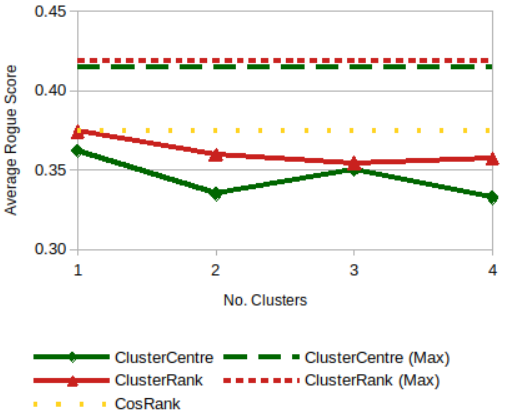

Fig. 7. Sensitivity of accuracy to the number of clusters

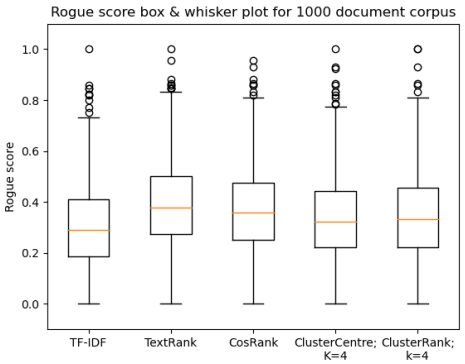

Fig. 8. Variability of accuracy over 1000 document corpus

4) Interpretation of Accuracy

The average accuracy for each method presented in this work are generally in the range 0.3 to 0.4. Considered as error (1 - accuracy) this may seem relatively high: 0.6 to 0.7. However, such values are in line with those reported in literature for similar methods. For example [4] reports a ROGUE-1 score of 0.47 for the basic TextRank method. While this is higher than the values reporter in this work, it was applied on a different data set (DUC, 2002) and may not be directly comparable. A comparable set of results on the Daily Mail data set is presented by [14], where ROGUE-1 scores of ~0.4 are reported for similar (extractive) document summarization techniques.

V. Conclusions and Future Work

The five text summarization methods considered in this work may be described as follows:

- TF-IDF: this is a basic and widely used approach which ranks sentences based on the frequency with which their terms appear in the document, and the inverse of the frequency with which they appear in documents comprising the corpus;
- TextRank: this implementation follows that described by [4]. It may be considered as the basic application of the TextRank algorithm to the case of text summarization. Sentence similarity was evaluated based on Jaccard similarity;
- CosRank: this method is an advance on the basic TextRank method in that it evaluates sentence similarity based on cosine similarity of embedded sentence vectors;

- ClusterCenter: this method does not employ the TextRank algorithm. Rather sentence vectors were clustered and ranked based on proximity to the centre of the clusters;
- ClusterRank: this method clusters embedded sentence vectors and applies the TextRank algorithm to each cluster separately.

A. General Conclusions

The following general conclusions relative to research questions are drawn:

1) How does the performance of a graph based text summarization method (TextRank) compare to a simpler method?

This work showed that text summarization methods employing the TextRank algorithm (TextRank, CosRank, ClusterRank) exhibited improved accuracy relative to those that did not (TF-IDF, ClusterCenter). Furthermore, the basic TextRank algorithm showed an order of magnitude lower computational demand relative to all other methods considered.

2) Can improvements to the graph based method be identified?

Sentence embedding did not deliver significant improvement in accuracy relative to the basic TextRank method using Jaccard similarity between sentences. Furthermore, sentence embedding resulted in significantly increased computational demand.

Unsupervised classification of embedded sentence vectors did not show improved accuracy relative to the basic TextRank method whether the TextRank algorithm was employed (ClusterRank) or not (ClusterCenter).

B. Specific Conclusions

The following more detailed conclusions are also noted:

The three methods that employed the TextRank algorithm to rank sentences (referred to in this work as TextRank, CosRank, and ClusterRank) had higher accuracy than the two methods that did not employ it (TF-IDF, ClusterCenter). This was consistent across the range of corpus sizes considered (10, 100, and 1000 documents).

The TextRank algorithm delivered improved accuracy relative to the simpler term frequency based approach (TF-IDF). For a corpus of 1000 documents the average Rogue score for TextRank was 0.39, compared to 0.31 for the TF-IDF method.

More advanced analysis of sentences using sentence embeddings did not deliver a significant increase in accuracy. For example, the results for the TextRank and CosRank methods were 0.39 and 0.37 respectively, with 1000 documents considered.

More advanced application of the TextRank algorithm by unsupervised classification of sentence vectors did not deliver significant improvement in accuracy. This was evident as the accuracy of the ClusterRank method deteriorated as cluster size increased.

Application of the TextRank algorithm to clustered sentence vectors (ClusterRank) delivered a modest improvement on accuracy relative to the case where clustered vectors were ranked based on proximity to the cluster center (ClusterCenter).

The two methods that employed unsupervised classification of sentence vectors were not optimized for cluster size. Rather, a single cluster size was applied over all documents. This may have contributed to the observed deterioration in accuracy as cluster size increased. Analysis of accuracy across a range of cluster sizes indicated that improved accuracy may be possible by optimizing cluster size for each document.

The TextRank method was an order of magnitude less computationally demanding than all other methods. For example, when applied to a 1000 document corpus its execution time was ~2 minutes compared to ~30 minutes for each of the other four methods.

The average summary size produced by all methods other than TF-IDF was larger (in terms of number of words) relative to the reference summary. This meant that TF-IDF was the only method that produced shorter summaries (on average) than the reference summary.

C. Future Work

The following areas of future work are noted:

The example qualitative assessment presented in this work showed that duplicate sentences that appeared in the original text were in some cases duplicated in the summaries. Methods of preventing such redundancy in the summaries should be explored. For example, this may be addressed by imposing a threshold maximum similarity between sentences such that identical sentences are not duplicated in the summary.

The methods presented in this work were restricted to unigrams. It would be interesting to extend this work to n-grams for $n > 1$.

Unsupervised classification was implemented according to a basic implementation of the k-Nearest Neighbours algorithm. Furthermore, constant values of k were employed over all documents in the corpus. Analysis of the sensitivity of accuracy to k indicated a potential improvement in accuracy if k was optimized on a document by document basis. Therefore it would be interesting to implement automatic identification of optimal cluster number.

The data set employed considered relatively short documents (news articles). It would be interesting to apply the methods to longer documents. In particular, this may be expected to leverage the benefits of unsupervised classification methods more effectively, since longer documents may contain more natural clustering of data associated with different topics.

It would be interesting to explore alternative methods of unsupervised classification. For example, “soft” clustering methods such as Gaussian Mixture Models may provide a more effective method relative to the “hard” clustering method associated with methods such as k-Nearest Neighbours.

References

[1] *Radev, D.R., Hovy, E. and McKeown, K., 2002. Introduction to the special issue on summarization. Computational linguistics, 28(4), pp.399-408.*

[2] *Liu, L., Lu, Y., Yang, M., Qu, Q., Zhu, J. and Li, H., 2018, April. Generative adversarial network for abstractive text summarization. In Proceedings of the AAAI Conference on Artificial Intelligence (Vol. 32, No. 1).*

[3] *Mihalcea, R., 2004, July. Graph-based ranking algorithms for sentence extraction, applied to text summarization. In Proceedings of the ACL interactive poster and demonstration sessions (pp. 170-173).*

[4] *Mihalcea, R. and Tarau, P., 2004, July. Textrank: Bringing order into text. In Proceedings of the 2004 conference on empirical methods in natural language processing (pp. 404-411).*

[5] *Allahyari, M., Pouriyeh, S., Assefi, M., Safaei, S., Trippe, E.D., Gutierrez, J.B. and Kochut, K., 2017. Text summarization techniques: a brief survey. arXiv preprint arXiv:1707.02268.*

[6] *Padmakumar, A. and Saran, A., 2016. Unsupervised text summarization using sentence embeddings. Dept. Comput. Sci., Univ. Texas, Austin, USA, Tech. Rep.[Online]. Available: https://www. cs. utexas. edu/~ aish/ut/NLPProject. pdf.*

[7] *Thakar, M., 2019. Comparing Text Summarization Techniques. URL: https://towardsdatascience.com/comparing-text-summarization-techniques-d1e2e465584e.*

[8] *Chauhan, K., 2018. Unsupervised Text Summarization using Sentence Embeddings. URL: https://medium.com/jatana/unsupervised-text-summarization-using-sentence-embeddings-adb15ce83db1.*

[9] *Hermann, K.M., Kočiský, T., Grefenstette, E., Espeholt, L., Kay, W., Suleyman, M. and Blunsom, P., 2015. Teaching machines to read and comprehend. arXiv preprint arXiv:1506.03340.*

[10] *Cho, K. DeepMind Q&A Dataset, URL: https://cs.nyu.edu/~kcho/DMQA/.*

[11] *Reimers, N. and Gurevych, I., 2019. Sentence-bert: Sentence embeddings using siamese bert-networks. arXiv preprint arXiv:1908.10084.*

[12] *SentenceTransformers Documentation. URL: https://www.sbert.net/*

[13] *Lin, C.Y., 2004, July. Rouge: A package for automatic evaluation of summaries. In Text summarization branches out (pp. 74-81).*

[14] *Nallapati, R., Zhai, F. and Zhou, B., 2017, February. Summarunner: A recurrent neural network based sequence model for extractive summarization of documents. In Proceedings of the AAAI Conference on Artificial Intelligence (Vol. 31, No. 1).*

Appendix A.

TABLE II. SAMPLE SUMMARY

Raw Text 7dc5574d1faac44b8f509b87cd9ffea90cbe892f.story File number: 8	A riot erupted at a Groundhog Day party in Russia after guests realised their host was planning to cook a marmot - and then serve it with cranberry sauce. The showbiz bash - held in Moscow - featured the marmot, which had been brought in as 'the star of the show' from a local children's petting zoo. A host of Russian celebrities happily posed with the creature, a relative of the squirrel, as they arrived at the charity event in aid of underprivileged children. Scroll down for video. A riot erupted at a themed Groundhog Day party in Russia after guests realised their host was planning to cook a marmot - and then serve it with cranberry sauce. A host of Russian celebrities happily posed with the creature, a relative of the squirrel, as they arrived at the charity event in aid of underprivileged children. The showbiz bash - held in the Russian capital Moscow - featured the marmot, which had been brought in as 'the star of the show' from a local children's petting zoo. But the guests were left with a very bad taste in their mouths when organiser Aleksey Polihun, 35, announced that he was about to kill and then cook the groundhog before serving it up on platter in a cranberry sauce. He said he wanted to kill and cook it as he thought it would be 'entertaining to have a groundhog which didn't survive groundhog day'. Groundhog Day is traiditonally held on February 2, when legend has it that if the rodent sees his shadow, winter will last another six weeks. If not, spring comes early. It is not clear why the party was being held so long after the official Groundhog Day. Guest Marya Nekrasova, 26, said: 'It was outrageous. 'The poor thing was terrified... and it may be hard to believe but some people actually cheered him on.' But the guests were left with a very bad taste in their mouths when organiser Aleksey Polihun, 35, announced that he was about to kill and then cook the groundhog before serving it up on platter in a cranberry sauce. He said he wanted to kill and cook it as he thought it would be 'entertaining to have a groundhog which didn't survive groundhog day'. The groundhog day special cake. One of the guests in the end offered to buy the animal off the menu and took it back to the zoo. She added: 'One of the other guests in the end offered to buy it off the menu and took it back to the zoo.' In the hit Groundhog Day movie, star Bill Murray is forced to live the same 24 hours over and over again while he is in a small town to report on a marmot said to be able to predict the future. Polihun said later: 'It's a pity. I think it would have been very tasty. 'A bit like rabbit but with more zing to it.' He added: 'In the end most people thought I should let him live, so I did. 'I just thought it would be entertaining to have a groundhog who didn't actually survive groundhog day.'
Reference Summary	Showbiz party held in honour of a children's charity and featured a marmot. Russian celebrities happily posed with the creature as they arrived at event. But riot erupted when host announced he was to 'kill, cook and serve' it. Host said it would be 'entertaining to have a groundhog which didn't survive groundhog day'. The animal survived the ordeal and is now at zoo.
TF-IDF Rogue = 0.27	He said he wanted to kill and cook it as he thought it would be 'entertaining to have a groundhog which didn't survive groundhog day'.He said he wanted to kill and cook it as he thought it would be 'entertaining to have a groundhog which didn't survive groundhog day' .The groundhog day special cake.'I just thought it would be entertaining to have a groundhog who didn't actually survive groundhog day.'
TextRank Rogue = 0.45	A riot erupted at a .Groundhog Day party in Russia after guests realised their host was planning to cook a marmot - and then serve it with cranberry sauce.A riot erupted at a themed Groundhog Day party in Russia after guests realised their host was planning to cook a marmot - and then serve it with cranberry sauce .He said he wanted to kill and cook it as he thought it would be 'entertaining to have a groundhog which didn't survive groundhog day'.He said he wanted to kill and cook it as he thought it would be 'entertaining to have a groundhog which didn't survive groundhog day' .
CosRank Rogue = 0.48	A riot erupted at a .Groundhog Day party in Russia after guests realised their host was planning to cook a marmot - and then serve it with cranberry sauce.A riot erupted at a themed Groundhog Day party in Russia after guests realised their host was planning to cook a marmot - and then serve it with cranberry sauce .He said he wanted to kill and cook it as he thought it would be 'entertaining to have a groundhog which didn't survive groundhog day'.But the guests were left with a very bad taste in their mouths when organiser Aleksey Polihun, 35, announced that he was about to kill and then cook the groundhog before serving it up on platter in a cranberry sauce .
ClusterCenter Rogue = 0.33	'The poor thing was terrified... and it may be hard to believe but some people actually cheered him on.'He said he wanted to kill and cook it as he thought it would be 'entertaining to have a groundhog which didn't survive groundhog day'.He said he wanted to kill and cook it as he thought it would be 'entertaining to have a groundhog which didn't survive groundhog day' .One of the guests in the end offered to buy the animal off the menu and took it back to the zoo .
ClusterRank Rogue = 0.67	A host of Russian celebrities happily posed with the creature, a relative of the squirrel, as they arrived at the charity event in aid of underprivileged children .A riot erupted at a .Groundhog Day party in Russia after guests realised their host was planning to cook a marmot - and then serve it with cranberry sauce.A riot erupted at a themed Groundhog Day party in Russia after guests realised their host was planning to cook a marmot - and then serve it with cranberry sauce .The showbiz bash - held in Moscow - featured the marmot, which had been brought in as 'the star of the show' from a local children's petting zoo.