News to Tweet: An Extractive Approach to Tweet Summary Generation

Ismail Mustafa
Courant Institute of
Mathematical Sciences
New York University
ismailmustafa@nyu.edu

Pranav Dhar
Courant Institue of
Mathematical Sciences
New York University
pranav.dhar@nyu.edu

Honor Pledge I pledge my honor that all the work described in this report is solely mine and that I have given credit to all third party resources that I have used.

Abstract

This paper discusses a method to convert news articles to summaries of less than 140 characters so that it can fit in a tweet. Previous studies have incorporated supervised learning algorithms to examine tweets based on news articles. We propose an extractive summarization algorithm called ”rank interpolation” which uses three extractive summary algorithms, namely LexRank, PageRank, and LSA, to produce more accurate results. We then implement a simple method of paraphrasal sentence compression in order to shorten the most highly ranked sentence to below 140 characters so that it can fit in a tweet. We find that our approach improves the summary accuracy markedly when compared to using each of the individual algorithms alone.

1 Introduction

1.1 Background

According to Eduard Hovy in ”The Oxford Handbook of Computational Linguistics” (Mitkov et al., 2003), a summary is defined as:

”...a text that is produced from one or more texts, that contains a significant portion of the information in the original text(s), and that is no longer than half of the original text(s).”

The salient points of this definition are that a summary should be half the length of the original and it should reflect the general idea of the text it summarizes. There are two main types of summarization techniques, extractive and abstractive. Extractive models identify and concatenate important sentences in the text to produce a more concise version of the original article.

Abstractive models on the other hand involve using context specific information to retain the main ideas of the article. Abstractive summarization is a very challenging problem as it attempts to mimic the way a human would generate a summary (Mitkov et al., 2003).

In addition to these approaches to summarization, there are two types of summaries that can be generated. The first is an indicative summary which is used to help the reader decide whether or not they want to read an article. The second type is an informative summary which serves as a condensed replacement of the article. Furthermore, the general task of automatic summarization can be split into single document or multi-document summarization (Hovy et al., 1998).

1.2 Problem Definition

With the ever increasing amount of news available from on-line sources, it can be overwhelming to get through all the news one needs in order to stay up to date with current events. In addition, studies conducted by the Pew Research Center show that 52% of all users on twitter rely on it as their primary source of news (Holcomb et al., 2013). Taking this into consideration users need to quickly be able to read a tweet and understand the gist of the article. Current methods of extractive summarization techniques produce good results but exceed the 140 character limit set by twitter.

In this paper, we develop an approach that uses three popular sentence ranking algorithms, namely Latent Sematic Analysis (LSA), LexRank, and PageRank in conjunction with linear interpolation

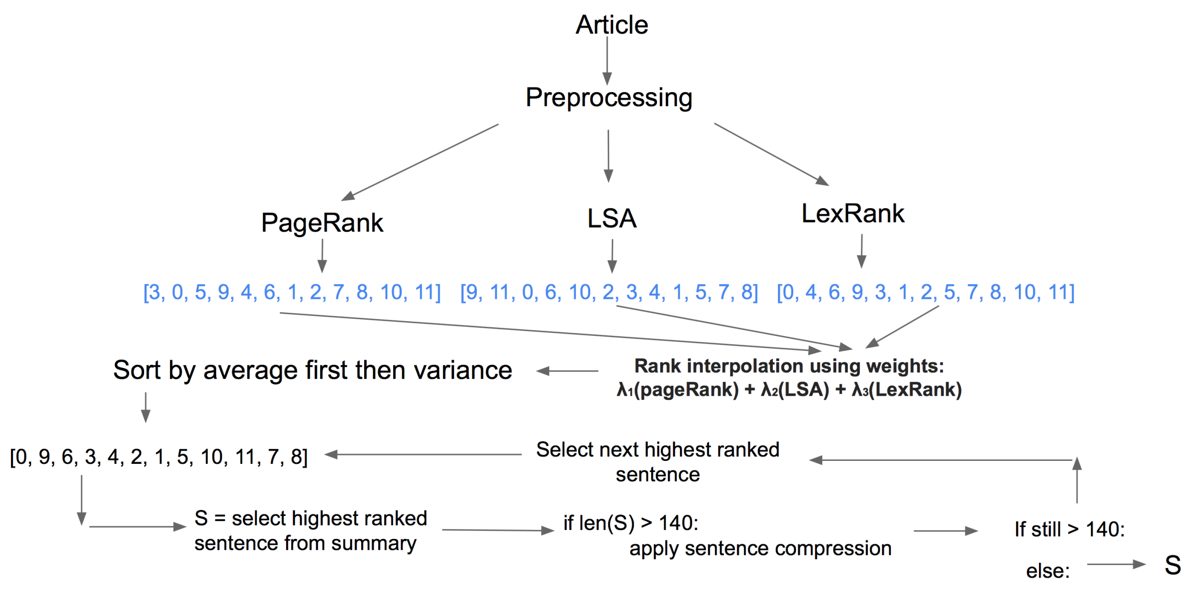
Article
Preprocessing
PageRank
LSA
LexRank
[3, 0, 5, 9, 4, 6, 1, 2, 7, 8, 10, 11]
[9, 11, 0, 6, 10, 2, 3, 4, 1, 5, 7, 8]
[0, 4, 6, 9, 3, 1, 2, 5, 7, 8, 10, 11]
Sort by average first then variance
Rank interpolation using weights: λ1(pageRank) + λ2(LSA) + λ3(LexRank)
[0, 9, 6, 3, 4, 2, 1, 5, 10, 11, 7, 8]
Select next highest ranked sentence
S = select highest ranked sentence from summary
if len(S) > 140: apply sentence compression
If still > 140:
else:
S

Figure 1: Schematic diagram of the Rank interpolation with Sentence Compression approach.

to determine the single most important sentence to display in the tweet. We then apply a simple paraphrasal sentence compression algorithm using the paraphrase database (Ganitkevitch et al., 2013) to ensure the sentence is shortened to less than 140 characters.

2 Related Work

There are a wide array of studies that address the problem of single and multi-document automatic summarization. Graph based approaches like Google's PageRank (Brin et al., 1998),(Kenai,) have been successfully used in social networks and the Web to analyse their link-structures. We employ a graph-based approach similar to TextRank (Rada Mihalcea et al.,) to perform automatic summarization. Another graph based approach that we investigated was proposed by (Güneş Erkan et al.,) "LexRank, for computing sentence importance based on the concept of eigenvector centrality in a graph representation of sentences." A common aspect of such graph based approaches are that the vertices represent the sentences and the edges are inter-sentence connections based on a similarity function.

However with approaches like PageRank and LexRank there is a loss of semantic information from the document, hence we incorporate a method that uses Latent Semantic Analysis as proposed by (Josef Steinberger, 2007) to obtain a representation of document topics. Although the aforementioned approaches give us a sentence that embodies the important aspects of the document, we have to make sure that the 140 character limit for a tweet is met. The paraphrase database (Ganitkevitch et al., 2013) along with the KenLM language model toolkit (Kenneth Heafield,) provides us with a reasonable method to shorten single sentences while retaining the most important information in them.

3 Approach

A schematic diagram of the approach we use is shown in Figure 1. The data we use for our evaluation is from the (DUC, 2002) corpus, document sets (061-120). Each document set contains multiple newswire/newspaper articles on a single subject. For each corresponding article we used single document abstracts (Type = "PERDOC", Summarizer="B", Length=100) as our reference for ROUGE (Chin-Yew Lin,) where we look at the F1 score as our primary evaluation metric. The DUC corpus contains abstracts of length 10, 50, and 100 words, however we were unable to use them as they were generated using multi-document summarization. We processed the raw corpus to get 567 newswire/news articles which we split into training data, containing 80% of the articles, and test data containing the rest.

In the final step of our pipeline we perform sentence compression, where use UPenn's 6 million word paraphrase corpus (Ganitkevitch et al., 2013). To ensure grammatically coherent sentences, we use the English Gigaword corpus which provides us with a trigram language model using modified Knesser-Nay smoothing (Keith Vertanen,).

3.1 Preprocessing

The first step in our implementation is to preprocess the news article. We tokenize the raw text into sentences. We then convert each word to lowercase, strip any html or xml tags, remove punctuation, strip multiple whitespaces, remove any numbers, remove stop words, remove words that are less than or equal to 3 characters in length, and apply a porter stemming to resulting text.

3.2 PageRank

The first algorithm in our pipeline uses the weighted graph representation of the processed document and the PageRank algorithm to compute the ranks of the vertices. In our implementation, we first need to build a graph where the vertices represent the sentences in the document. Next we need to compute the term frequency as follows:

$$tf_{i,j} = \frac{freq_{i,j}}{max_l freq_{l,j}}$$

where we normalize using the maximum frequency in a given sentence and the inverse sentence frequencies as follows:

$$isf_i = log\frac{N}{n_i}$$

where N is the number of sentences and n_i is the number of sentences in which the term occurs. Using the tf-idf matrix we can then compute the edges using the cosine similarity function as shown below:

$$W(s_m, s_n) = \frac{\sum_{i=1}^{t} w_{i,m} w_{i,n}}{\sqrt{\sum_{i=1}^{t} w_{i,m}^2}\sqrt{\sum_{i=1}^{t} w_{i,n}^2}}$$

Finally we run the ranking algorithm on the graph based on the recursive function shown below:

$$PR(V_i) = (1-d) + d \sum_{V_j \epsilon In(V_i)} \frac{PR(V_j)}{|Out(V_j)|}$$

Once the ranking algorithm converges to a set of scores for our graph, we assign a rank to each sentence in reverse order of their score ie. the sentence with the highest score will have a rank of 1.

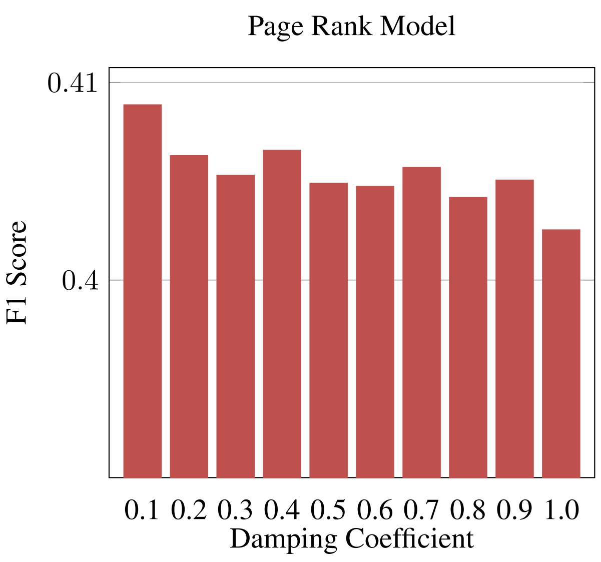

Figure 2: Damping Factor vs. F1 Score on training Data.

Evaluation: We ran PageRank on the training data changing the damping factor in incremental steps of 0.1. From the results summarized in Figure 2 we observe that a damping factor of 0.1 yields the highest F1 score. As we increase the damping factor, the component contributed by the neighbouring nodes in the PageRank equation increases. This could explain why our F1 store increases as highly connected sentences are ranked higher.

3.3 LexRank

The LexRank algorithm as proposed by (Güneş Erkan et al.,) is a slight modification of the aforementioned PageRank algorithm. In LexRank the notion of centrality is introduced where the main hypothesis states that "sentences that are similar to many of the other sentences in a cluster are more central (or salient) to the topic." The initial steps of the algorithm are identical to PageRank except for the modified similarity equation shown below:

$$W_{lex}(x,y) = \frac{\sum_{w \in x,y} tf_{w,x} tf_{w,y} idf_w^2}{\sqrt{\sum_{x_i \in x}(tf_{x_i,x} idf_{x_i})^2} \times \sqrt{\sum_{y_i \in y}(tf_{y_i,y} idf_{y_i})^2}}$$

The process of computing LexRank scores is described in the psuedo-code shown in Figure 3 where we can see the computation of the degree vector using the threshold value of t.

Evaluation: Similar to PageRank we determined a good damping ratio, d = 0.25. We then ran the algorithm on the training set varying the threshold values. From the results shown in Figure 4 we can see that the F1 Score plateaus beyond a threshold value of 0.15.

```
Input: An array S of n sentences,
       cosine threshold t
Output: An array L of LexRank scores

Array CosineMatrix[n][n]
Array Degree[n]
Array L[n];
for i <- 1 to n do
  for j <- 1 to n do
    CosineMatrix[i][j] - Wlex(S[i][j])
      if CosineMatrix[i][j] > t
        CosineMatrix[i][j] = 1
        Degree[i]++;
      else
        CosineMatrix[i][j] = 0
      end
  end
end
for i <- 1 to n do
  for j <- 1 to n do
    CosineMatrix[i][j] /= Degree[i];
  end
end
return PowerMethod(CosineMatrix,n,E)
```

Figure 3: Algorithm for computing LexRank scores.

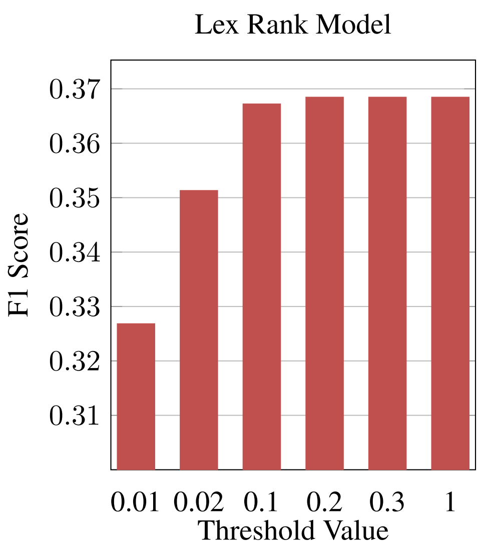

Figure 4: Threshold vs F1 score on Training Data (d = 0.25)

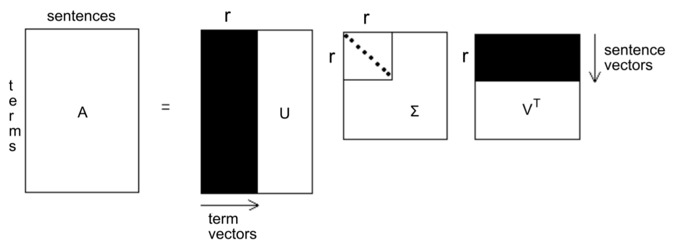

Figure 5: Singular Value Decomposition of Matrix A.

3.4 Latent Semantic Analysis

LSA is a useful technique to extract ”hidden dimensions” of the semantic representation of terms and sentences within a document. A suitable method for text summarization is proposed by (Josef Steinberger, 2007) with the following steps. First we create a matrix A as shown in Figure 5 where the rows and columns are the terms and sentences of the document respectively. For multi-document considerations we can also add an additional global weighting system taking into account terms across the training set.

In the next step we use Singular Value Decomposition to break down Matrix A into the following components:

$$A_{mn} = U_{mm} S_{mn} V^T_{nn}$$

The terms are the rows of the orthonormal vector U where the signs of the coefficients in each column indicate the co-occurrence patterns of the words and S represents the variance of the linearly independent components.

We use our training set to tune a parameter to reduce the dimensionality and compute

$$R_{mr} = A_{mr} S_{rr}$$

where $r = ReductionRatio * n$. This term is important in deciding how many LSA dimensions/topics we need to include in the latent space. If we select too few, we may lose important aspects of the document in the summary. In the next step we breakdown the matrix A into r linearly independent ”topics” which we can then use to cluster the terms and sentences on a semantic basis rather than just on the basis of word occurrence. Finally we want to rank sentences which have the greatest combined weight across r topics, hence we compute matrix B as follows:

$$B = S^2.V^T$$

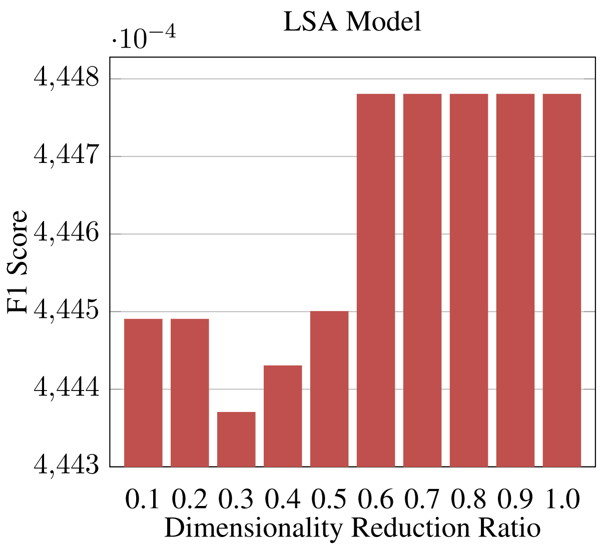

Figure 6: Reduction Ratio vs F1 score on Training Data

and measure the length for each k^{th} sentence using:

$$s_k = \sqrt{\sum_{i=1}^{r} b_{i,k}^2}$$

Evaluation: We ran LSA on the training data changing the dimensionality reduction ratio in incremental steps of 0.1. From the results summarized in Figure 6 we can infer that there is a drop in the F1 score when the number of dimensions are reduced by more than 40%. This matches our intuition that a reduction in the dimension of r causes us to lose important topical information from the document.

3.5 Rank Interpolation

This is the final step for determining the sentence ranks. We perform linear interpolation on the 3 rank vectors returned by the PageRank, LexRank & LSA algorithms. We tune the λ parameters on the training set by running the system over various combinations in step sizes of 0.1. The 3D scatter plot shown in Figure 7 represents the F1 score for the various combinations of λ_1, λ_2 & λ_3. The points are color coded where the yellow end of the spectrum corresponds to the highest F1 score value generated by our system.

Evaluation: From the results shown in Figure 7, we get a maximum F1 score of 0.47866 for $\lambda_1 = 0.4$, $\lambda_2 = 0.3$ & $\lambda_3 = 0.3$.

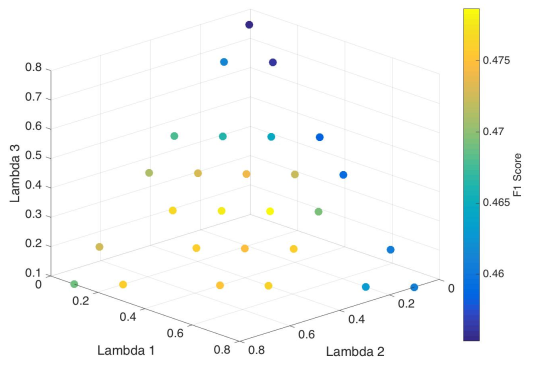

Figure 7: F1 score of rank interpolation method as a function of λ_1, λ_2, and λ_3

3.6 Sentence Compression

Once we use rank interpolation to pick the most relevant sentence from the news source, we split the sentence into every possible combination of its parts. As an example we take the sentence ”Today it is sunny”. The resulting set of every combination of its parts would be as follows:

```
['today', 'it is sunny']
['today it', 'is sunny']
['today it is', 'sunny']
['today it is sunny']
['today', 'it', 'is sunny']
['today', 'it is', 'sunny']
['today', 'it is sunny']
['today', 'it', 'is', 'sunny']
['today', 'it', 'is sunny']
['today', 'it', 'is', 'sunny']
```

We then take UPenn’s paraphrase database (Ganitkevitch et al., 2013) where each line of the 6 million entries is formatted as such:

```
... ||| PHRASE ||| PARAPHRASE ||| ...
```

We filter the entries such that PARAPHRASE must be shorter than PHRASE. This ensures that all paraphrased guesses will result in a shorter sentence. We then query the parphrase database for each phrase in all of the possible combinations. Of these paraphased lists, we only keep the result if it does not shorten the list by any more than 30% of the original length. This limit of 30% compression was heuristically determined to produce more coherent results. The paraphrased outputs then becomes:

```
['', 'it is sunny']
['today it', 'is sunny']
['today it is', 'sunny']
['today it is sunny']
```

We then concatenate all potential paraphrased lists back into sentences and score the sentence using the KenLM language model toolkit (Kenneth Heafield,) along with a trigram Knesser-Nayes smoothed language model created from the English Gigaword Corpus (Keith Vertanen,). The scores are determined by averaging the negative log trigram probabilities of each sentence according to the language model. The scores for each potential paraphrased sentence are shown below:

```
it is sunny         11.2157001495
today it is sunny   14.1172990799
today it is sunny   14.1172990799
today it is sunny   14.1172990799
```

Picking the lowest score gives us the shortest sentence that is likely to be grammatically correct as per the language model which in this case results in the sentence ”it is sunny”. The temporal word ”today” was dropped as this was one of the rules according to the paraphrase database. This method of sentence compression is applied to the highest ranked sentence given by the rank interpolation algorithm as long as the sentence is greater than 140 characters. If it fails to compress it below 140 characters, it picks the next best sentence and tries again. This continues until all possibilities are exhausted or a sentence is found to be less than 140 characters.

3.7 Results

Our results show the precision, recall, and F1 scores for all four algorithms for both the training and test data in Table 1 and Table 2 respectively. Additionally, we include an example of the algorithm summarizing an article to fit in a tweet in Table 3. In the test data results of Table 2, of the three individual algorithms, LSA performed the best yielding an F1 score of 0.45908. As per Josef Steinberger's PhD thesis reagarding LSA, his training on the same data set yielded an F1 score of 0.42776 (Josef Steinberger, 2007). Our score is slightly higher since we limit our summary to one sentence when evaluating on reference summaries. Our LexRank is roughly in line with Güneş Erkan et al. (Güneş Erkan et al.,) as their F1 score of 0.4168 is almost the same as our F1 score of 0.41967. Our PageRank algorithm performs slightly better than (Kenai,) which got an F1 score of 0.3829 compared to our 0.43323. Again this could be due to the fact that ROUGE evaluation is done on only one sentence. Finally, our Rank Interpolation algorithm performed the best, surpassing the other algorithms by a noticeable amount with an F1 score of 0.49107.

In Table 3, we apply Rank Interpolation to the sample article displayed. The interpolation algorithm extracts the first sentence of the article as it is determined to be the most important:

”The government today imposed direct federal rule in Kashmir, where Indian security forces are battling to put down a rebellion by Moslem separatists.”

This sentence has a character count of 149 characters and is unfit for use in a tweet. We then apply sentence compression to get the result shown in Table 3 which is a sentence that contains 139 characters, short enough to fit in a tweet. The sentence compression algorithm removed the temporal word ”today”, and replaced government with public.

Running the sentence compression algorithm on 500 sentences, we find the compression percentage to be 5.99% on average which is a testament to how conservative the algorithm we implement is.

Model	**Precision**	**Recall**	**F1 Score**
PageRank	0.4802	0.36674	0.40887
LexRank	0.55361	0.28324	0.36844
LSA	0.38551	0.53521	0.44478
Rank Intrpl	0.42476	0.56074	0.47866

Table 1: Precision, Recall, and F1 scores for training data

Model	**Precision**	**Recall**	**F1 Score**
PageRank	0.50872	0.38608	0.43323
LexRank	0.60784	0.32737	0.41967
LSA	0.38825	0.57056	0.45908
Rank Intrpl	0.42485	0.58992	0.49107

Table 2: Precision, Recall, and F1 scores for test data

4 Conclusion and Future Work

This paper looked into a method to convert news articles to summaries of less than 140 characters for use in tweets. We used an approach we called ”rank interpolation” which uses three extractive

Article	Result
The government today imposed direct federal rule in Kashmir, where Indian security forces are battling to put down a rebellion by Moslem separatists. The action follows the killing Tuesday of at least 29 Moslem militants by Indian security forces. In the Pakistani capital of Islamabad, meanwhile, India and Pakistan opened negotiations today aimed at repairing relations threatened by the rebellion in the disputed northern region. Relations between Pakistan and India worsened in January when Indian soldiers cracked down on Moslem militants in Kashmir who are seeking independence from India or union with Pakistan. At least 792 people have been killed in Kashmir since then. India has accused Pakistan of arming and training the militants, a charge Pakistan has denied.	The public imposed direct federal rule in Kashmir, where Indian security forces are battling to put down a rebellion by Moslem separatists.

Table 3: Result of ranked interpolation and sentence compression on a sample article

summary algorithms, LexRank, PageRank, and LSA, to produce a more accurate summary. We then used paraphrasal sentence compression in order to compress the selected sentence to below 140 characters. We found that our approach improved accuracy markedly when compared to the results of the indidual algorithms implemented alone.

In terms of future work, we would like to investigate sentential feature extraction as proposed by (Kupiec, Pedersen,). Using features extracted from our training data, we could improve our weighting system used in our PageRank and LexRank approaches. We compute the probability of a word s from the source text present in the summary as follows:

$$P(s\epsilon S|F_1...F_k) = \frac{P(F_1...F_k|s\epsilon S)P(s\epsilon S)}{P(F_1...F_k)}$$

Where F represents features such as:

1. Sentence length (words > 5)
2. Paragraph count
3. Paragraph feature (position in the document)
4. Uppercase feature

Other sentence similarity measures we would like to investigate in the future are making use of string kernels and longest common sub-sequences in order to evaluate their impact on the summarization performance. We would also like to explore a better approach to sentence compression using techniques that involve pruning dependency parse trees. This is because para-phrasal techniques sometimes result in changing the meaning of the sentence since contextual comprehension isn't taken into account.

Acknowledgments

We are grateful to our Professor, Slav Petrov for his guidance and advice during the project. We would also like to thank our guest lecturers David Weiss, Emily Pitler, Dipanjan Das, Tom Kwiatkowski, and Kuzman Ganchev.

References

Zhongyu Wei, Yang Liu, Chen Li, Wei Gao *Using Tweets to Help Sentence Compression for News Highlights Generation* Computer Science Department, The University of Texas at Dallas Richardson, Texas 75080, USA, Qatar Computing Research Institute, Hamad Bin Khalifa University, Doha, Qatar.

Brin and L. Page. 1998. The anatomy of a large-scale hypertextual Web search engine. Computer Networks and ISDN Systems

Rada Mihalcea and Paul Tarau.TextRank: Bringing Order into Texts Department of Computer Science University of North Texas

Josef Steinberger *Text Summarization within the LSA Framework* University of West Bohemia, January 26, 2007.

Güneş Erkan, Dragomir R. Radev. *LexRank: Graph-based Lexical Centrality as Salience in Text Summarization* University of Michigan Ann Arbor, Web. 18 Dec. 2015.

Das, Dipanjan, and Andre F.T. Martins. "A Survey on Automatic Text Summarization." (2007): n. pag. 21 Nov. 2007. Web. 12 Dec. 2015.

Mitkov, Ruslan. "Text Summarization." The Oxford Handbook of Computational Linguistics. Oxford: Oxford UP, 2003. 583-98. Print.

Hovy, Eduard, and Chin-Yew Lin. ”Automated Text Summarization and the SUMMARIST System.” Proceedings of a Workshop on Held at Baltimore, Maryland October 13-15, 1998 - (1996): n. pag. Web.

Holcomb, Jesse, Jeffrey Gottfried, and Amy Mitchell. ”News Use Across Social Media Platforms.” Pew Research Centers Journalism Project RSS. Pew Research Center, 14 Nov. 2013. Web. 8 Dec. 2015.

Ganitkevitch, Juri, Benjamin Van Durme, and Chris Collision-Burch. ”PPDB: The Paraphrase Database.” Proceedings of NAACL-HLT (2013): 758-64. Web. 13 Dec. 2015.

”Document Understanding Conferences.” Document Understanding Conferences. NIST, 09 Sept. 2014. Web. 16 Dec. 2015. ¡http://duc.nist.gov/¿.

Lin, Chin-Yew. ”ROUGE: A Package for Automatic Evaluation of Summaries.” (n.d.): n. pag. Web. 17 Dec. 2015. ¡http://www.aclweb.org/anthology/W04-1013¿.

Vertanen, Keith. ”English Gigaword Language Model Training Recipe.” English Gigaword Language Model. N.p., n.d. Web. 10 Dec. 2015. ¡http://www.keithv.com/software/giga/¿.

Heafield, Kenneth. ”KenLM Language Model Toolkit.” Kenlm . Code . Kenneth Heafield. N.p., n.d. Web. 03 Dec. 2015. ¡https://kheafield.com/code/kenlm/¿.

Kupiec, Pedersen, Chen: A trainable document summariser, SIGIR 1995

Kenai: A trainable document summariser, SIGIR 1995 https://kenai.com/downloads/textsummarizer/FinalWriteups.pdf