Text Summarization

Ankita Nallana
nallana.a@husky.neu.edu
NLP 6120 Project Report

Abstract

A graph based approach to Extractive Text Summarization by defining sentence centrality, computing LexRank scores and selecting the top ranked sentences for inclusion in a summary

Introduction

Summarization is the process of presenting important information from document(s) in a short, concise manner. This is a very handy task when we are presented with long documents such as scientific text, news articles, etc. A summary would be helpful in quickly going through the significant aspects without spending much time reading through all of the document especially relevant now since on a daily basis we wish consume a lot of information but have not enough time.

When we, i.e. humans, are asked to summarize a document or multiple documents, we read through the entire text and produce a summary in our own words. In the field of text summarization, this is referred to as *Abstractive Summarization*. *Abstractive Summarization* is when the entire document is read, its salient points are identified and summary abstracts are generated. This usually requires advanced language generation techniques to produce grammatical sentences.

Another approach to summarization is to pick best/most informative sentences from the document and present those, *verbatim*, as a summary. This is termed as *Extractive Summarization*. *Extractive summarization* is more concerned with the content that should be presented and many approaches focus on finding the right paradigms which would capture the essence of the document i.e. identify its important points. The distribution of information may not consistent throughout the document. For instance, the middle paragraphs might contain more relevant information than the first and last paragraph (which usually tend

to be more broad) which makes the problem more difficult – how are important sentences in a document identified?

Early Work

Work on extractive summarization dates as far back as 1958 and were carried out on magazine extracts and technical documents. Luhn in 1958[1] stated that the frequency of certain words provides a measure of a sentence's significance. These *significant words* were counted within a sentence which gave a *significance factor* to every sentence. All sentences were then ranked with respect to this factor and the top ranked *n* sentences made up the the summary. Baxendale, in 1958, explored the role of another feature – *sentence position* – in producing auto-extracts. Edmundson, in 1969[2], proposed another system where apart from *word frequency* and *sentence position*, *cue words* and *skeleton* (if sentence is present in a heading, title, and so on) were considered. Weights were assigned to sentences based on presence of these features and the sentences were accordingly scored.

Extractive summarization is also helpful in the case of *multi-document summarization*. The problem transforms into a slightly different one in this case. There can be content overlap (and perhaps even contradiction) across various documents. The problem is then selecting the most informative sentences from various sources and at the same time maintaining information which is coherent and covers all details.

Supervised Approaches

With the advent of Machine Learning techniques, various approaches were explored in this domain. In a supervised learning approach, one can 'learn' features of sentences that make them good candidates for inclusion in a summary. A classifier can be modeled such that it classifies whether a sentence should be '*abstract-worthy*' or not. However, in the dataset it must be known which sentences should be included and which ones should not be. This could be difficult (and/or expensive since a human has to mark sentences which may or may not be abstract-worthy) to obtain a dataset that is annotated this way. [*Note: I usually came across 'half' of these datasets – either just the documents or just the summaries*]

In a paper by Kupiec et al published in 1999, the authors investigated the use of a Bayesian classifier for summarizing text. Later on, Miles Osborne[3] compared a

MaxEnt model with a Naïve Bayes model - various features such as *sentence position*, *word-pairs*, *sentence-length*, etc have been experimented with using a *Naïve Bayes* model and a *Maximum Entropy* (*MaxEnt*) model. However, with the use of an 'Oracle' (which could correctly classify an unseen sentence) the *MaxEnt* model seemed to outperform the *Naïve Bayes* significantly.

Unsupervised Approaches

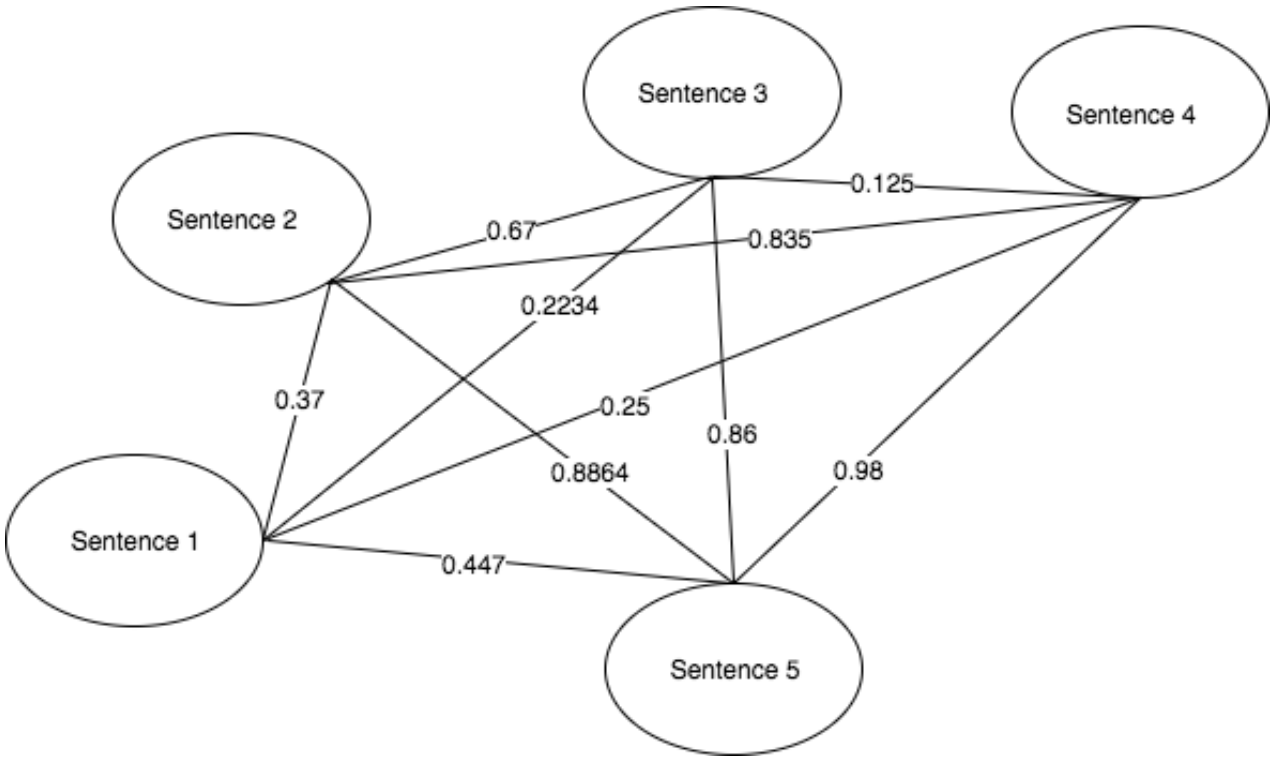
When training data as described above is difficult to obtain, an unsupervised approach to summarization can be useful. One approach to unsupervised learning is the "*Centroid*" approach (by Radev et al, 2000[4]) where a "*centroid*" or central sentences for a (multi-document) cluster of sentences are defined which capture the main theme of the cluster. Sentences which have more words in common with these centroids were considered 'central' or most important.

Soon after, in 2004, two similar methods, *TextRank* and *LexRank*, were developed which take on a graph based approach to the problem. The sentences in a document represent individual nodes in a graph. These nodes are connected with each other and their edges have certain weights assigned to them. These edges may convey information such as the similarity between two nodes (or sentences). The sentences are then ranked using the *PageRank* method (famously used by Google for their search engine) and the top *n* ranking sentences are picked to be included in the summary.

For my project on (extractive) Text Summarization, I chose the *LexRank* method – details of which are published in a paper by G. Erkan and D. Radev in 2004 in the *Journal of Artificial Intelligence* titled *LexRank : Graph-based Lexical Centrality as Salience in Text Summarization*

LexRank :

Erkan & Radev developed a new approach to Text Summarization – called '*LexRank'*, which is short for *lexical PageRank*. It uses the concept of eigenvector centrality in a graph – the graph here being a network of nodes where the nodes represent individual sentences in a document. The concept of eigenvector centrality comes into picture while determining the central sentences of a document.



Sample Representation of five sentences with their similarity scores in a document as a graph

In this approach, the 'centrality' of each sentence is calculated from a cluster of sentences and the most important ones are extracted to be included in the summary. Some sentences would be similar to some sentences and not very similar to other sentences (i.e. may have lesser information in common). The former would compose of some of the most 'central' sentences in a document.

This gives rise to two questions – how to define similarity between (any) two sentences and how to compute its centrality given its similarity to other sentences.

The similarity between sentences is calculated using the *idf-modified-cosine*:

$$\text{idf-modified-cosine}(x, y) = \frac{\sum_{w \in x,y} \text{tf}_{w,x} \text{tf}_{w,y} (\text{idf}_w)^2}{\sqrt{\sum_{x_i \in x} (\text{tf}_{x_i,x} \text{idf}_{x_i})^2} \times \sqrt{\sum_{y_i \in y} (\text{tf}_{y_i,y} \text{idf}_{y_i})^2}}$$

where :

w is a word in both x, y
x, y are the two sentences between which we are computing the similarity score

tf is the term-frequency of a word in the document
idf is the inverse-document-frequency of the word in the document

A similarity matrix is then constructed in which any two indices *i, j* correspond to sentences *i, j* in a document and an entry in the *[i][j]* cell of the matrix denotes the similarity between those two sentences.

Cosine similarity is useful measure for determining how '*similar*' two sentences are. If we represent sentences as vectors in a 2D space, two sentences which are very similar would subtend an angle of *0* degrees (either because they overlap or are parallel) and two sentences that are dissimilar would subtend an angle of *90* degrees. If two sentences are somewhat similar, they would subtend an angle between *0* and *90* degrees. The smaller the angle, the greater the similarity between the sentences.

To assess a sentence's overall centrality, we can count the number of sentences similar to it. In other words, the degree of the node corresponding to that sentence. The greater the degree, the more central a node(/sentence). *Degree Centrality* is calculated by considering cosine similarity scores above a certain *threshold*. Too low a threshold may mistakenly consider weak similarities and too high a threshold may lose too many similarity relations in a cluster. The resulting graph is less dense than the original since edges (or similarities) below the threshold are not considered.

When there are no threshold values to compare the similarity scores against then we have a much denser (but weighted graph). This is referred to as *c-LexRank* or *continuous LexRank*. The main idea behind *c-LexRank* is to observe the strength of the similarity links.

While assessing the *degree centrality* of a node, we consider each edge of a node as a *vote* to determine its centrality in the cluster. However, there can be a case where a many unwanted sentences vote for each other and raise their centrality. One method of dealing with this is to consider every node's centrality value and distributing it amongst it neighbors. To state it mathematically:

$$p(u) = \sum_{v \in adj[u]} \frac{p(v)}{deg(v)}$$

where $p(u)$ is the centrality of node u, $adj(u)$ are nodes adjacent to u and $deg(v)$ is the degree of each node v (belonging to $adj(u)$).

The similarity matrix is then run through a simple iterative method, also called the *power method*, which, using the concept of *PageRank* – where similar sentences would '*recommend'* similar sentences and raise its importance – returns the final scores or '*rating*'s of every sentence. This new measure of sentence centrality is referred to as the *LexRank* score.

Finally, the procedure as described in the paper :

1. Build a *Similarity Matrix* : where *similarityMatrix[i][j]* holds the cosine similarity value between sentences *i* and *j*
2. Assign *1.0* to *similarityMatrix[i][j]* if *similarityMatrix[i][j] > threshold_value* (if a threshold value is provided – not needed if *c-LexRank*)
3. Compute *LexRank* scores by *power iteration method*
4. Select *n* sentences with the highest score
5. Sort the sentences by original order picked in Step *4*
6. These sentences make up our summary abstract

Evaluation and Results :

In my implementation, I have followed the procedure exactly as stated above. I ran the program for a few news articles from the BBC dataset of varying length. I manually created summaries as a baseline to measure the generated summaries against. I set the summary length to *41%* of the entire document. I arrived at this figure by averaging over the lengths of manually created summaries.

Three summaries were generated per document – one by using *c-LexRank* and the other by using *LexRank* with a threshold of *0.1* and *0.2*

One of the methods of evaluation for generated summaries is comparing the precision and recall scores.

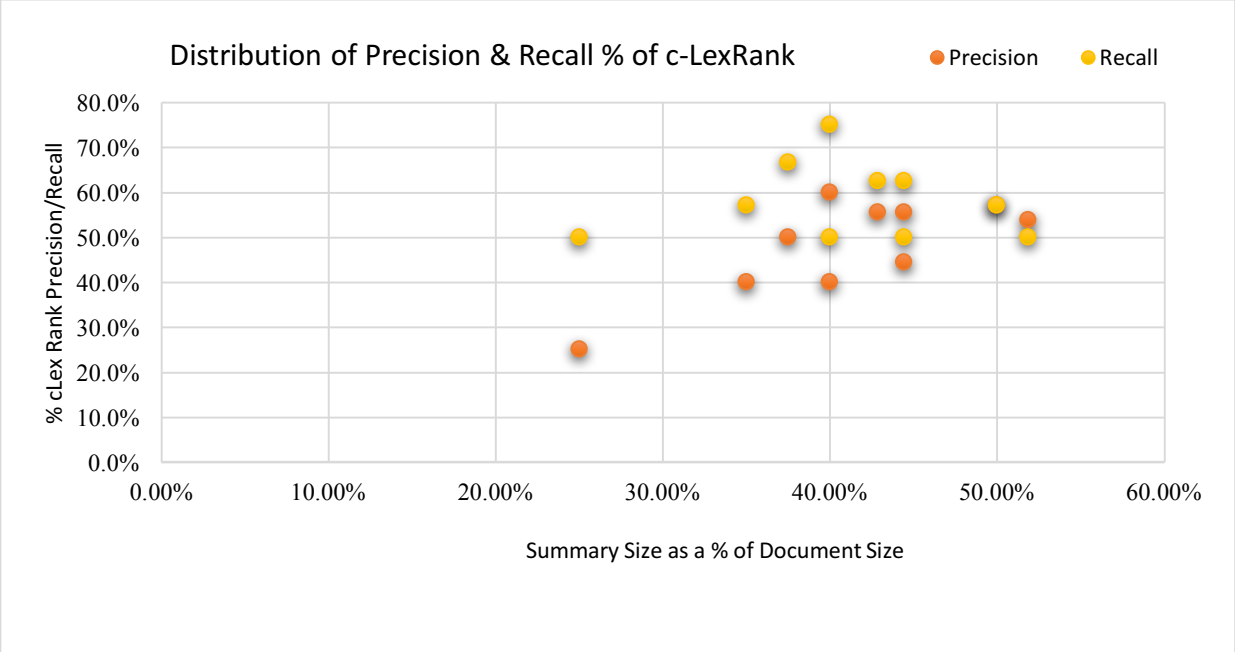
To calculate both scores, we require a *reference summary(R)* to compare our *generated summary(G)* against.

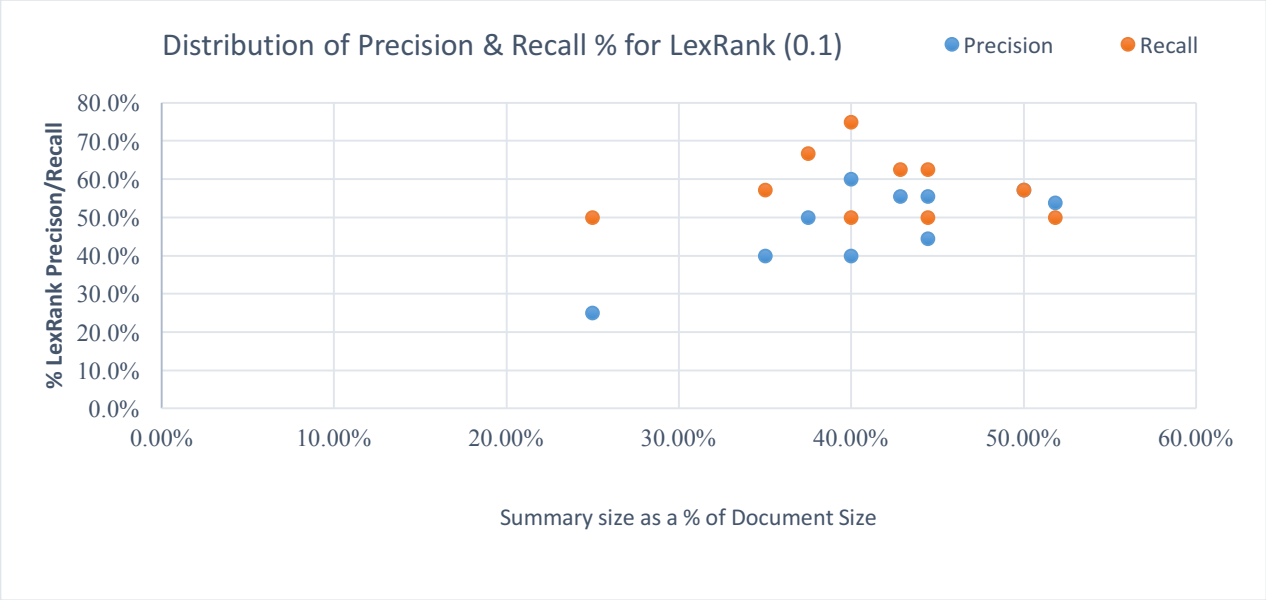
$$\text{Recall} = \frac{\text{intersection}(R, G)}{R}$$

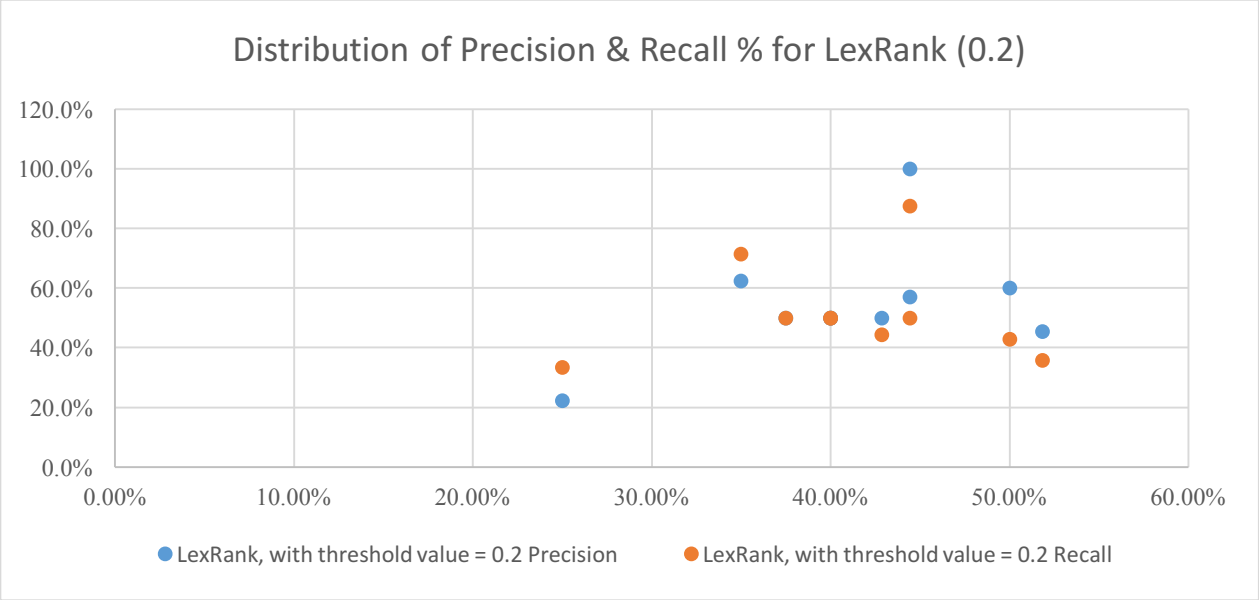
$$Precision = \frac{\text{intersection}(R, G)}{G}$$

Recall identifies what fraction of expected results is returned by the program And *precision* identifies what fraction of those results are correct/valid. I computed both *Recall & Precision* scores for all three summaries and present my observations as below :

File Name	LexRank (threshold value = 0.1)		LexRank (threshold value = 0.2)		c-LexRank	
	Precision	Recall	Precision	Recall	Precision	Recall
018.txt	60.0%	75.0%	50.0%	50.0%	40.0%	50.0%
025.txt	25.0%	50.0%	22.2%	33.3%	41.7%	83.3%
033.txt	50.0%	66.7%	50.0%	50.0%	50.0%	66.7%
037.txt	40.0%	50.0%	50.0%	50.0%	40.0%	50.0%
051.txt	40.0%	57.1%	62.5%	71.4%	50.0%	71.4%
058.txt	57.1%	57.1%	60.0%	42.9%	57.1%	57.1%
068.txt	55.6%	62.5%	100.0%	87.5%	44.4%	50.0%
077.txt	53.8%	50.0%	45.5%	35.7%	53.8%	50.0%
096.txt	44.4%	50.0%	57.1%	50.0%	55.6%	62.5%
104.txt	55.6%	62.5%	50.0%	44.4%	60.0%	60.7%







In all three cases, we observe that *Precision* values are around *50% – 60%* for summaries which are *35% - 45%* of the original length (with one summary reporting a score of 100%). The *Recall* values are higher than the *Precision* values for *c-LexRank* and *LexRank (0.1)* but the opposite is true for *LexRank (0.2)*

The average precision value for *LexRank (0.2)* is higher than that of *LexRank (0.1) & c-LexRank*. This might be due to the fact that a threshold of *0.2* acts as a *better* filter for picking the most informative sentences from the document. *C-LexRank* exhibited a good average recall of *60%* followed by *LexRank (0.1)* and *LexRank*

(0.2) – it almost suggests that a higher precision could be at a cost of lower recall. A higher threshold of *0.3* resulted in a dip in the precision values suggesting that perhaps in the process of picking out very similar sentences, a lot of other 'informative' sentences were missed.

These metrics are however referenced against a human-created summary which may or may not be ideal. The human mind conforms to biases and may assign some sentences with more importance than the rest. Even if the same document were given to two different humans to 'summarize' (or extract important sentences) – both summaries would be different and there's no definition of an ideal summary.

Sample Results:

File – *018.txt*
India's rupee hits five-year high India's rupee has hit a five-year high after Standard & Poor's (S&P) raised the country's foreign currency rating. The rupee climbed to 43.305 per US dollar on Thursday, up from a close of 43.41. The currency has gained almost 1% in the past three sessions. S&P, which rates borrowers' creditworthiness, lifted India's rating by one notch to 'BB+'. With Indian assets now seen as less of a gamble, more cash is expected to flow into its markets, buoying the rupee. "The upgrade is positive and basically people will use it as an excuse to come back to India," said Bhanu Baweja, a strategist at UBS. "Money has moved out from India in the first two or three weeks of January into other markets like Korea and Thailand and this upgrade should lead to a reversal". India's foreign currency rating is now one notch below investment grade, which starts at 'BBB-'. The increase has put it on the same level as Romania, Egypt and El Salvador, and one level below Russia.

Generated Summaries:

LexRank with 0.1 threshold
Indias rupee has hit a fiveyear high after Standard Poors SP raised the countrys foreign currency rating The upgrade is positive and basically people will use it as an excuse to come back to India said Bhanu Baweja a strategist at UBS Money has moved out from India in the first two or three weeks of January into other markets like Korea and Thailand and this upgrade should lead to a reversal The increase has put it on the same level as Romania Egypt and El Salvador and one level below Russia
LexRank with 0.2 threshold
Indias rupee has hit a fiveyear high after Standard Poors SP raised the countrys foreign currency rating The rupee climbed to 43305 per US dollar on Thursday up from a close of 4341 The currency has gained almost 1 in the past three sessions Indias foreign currency rating is now one notch below investment grade which starts at BBB

c-LexRank
Indias rupee has hit a fiveyear high after Standard Poors SP raised the countrys foreign currency rating With Indian assets now seen as less of a gamble more cash is expected to flow into its markets buoying the rupee Money has moved out from India in the first two or three weeks of January into other markets like Korea and Thailand and this upgrade should lead to a reversal Indias foreign currency rating is now one notch below investment grade which starts at BBB

Future Work

Although *LexRank* was originally developed to address *Multi-Document Summarization*, it performs well for *Single Document Summarization*. The same methodology is extended to *Multi-Document Summarization* for a certain topic except the number of sentences to process would be greater. However, there is an additional problem of information overlap for the most informative sentences. This could be handled by *Cross-Sentence Information Subsumption* (CSIS) where sentences that are too similar to each other in the summary are discarded.

Another approach to Text Summarization would be to implement one of the many Machine Learning techniques developed so far. However, the dataset for these are hard to come by (where documents and their summaries are annotated). I did come across just one such dataset but the summary extract-document size ratio was too small (4 sentences marked for a 50 sentence document, etc).

References :

[1] Luhn, H. P. (1958). The automatic creation of literature abstracts. IBM Journal of Research Development,

[2] Edmundson, H. P. (1969). New methods in automatic extracting. Journal of the ACM

[3] Osborne, M. (2002). Using maximum entropy for sentence extraction. In Proceedings of the ACL'02 Workshop on Automatic Summarization

[4] Radev, D. R., Jing, H., and Budzikowska, M. (2000). Centroid-based summarization of multiple documents: sentence extraction, utility-based evaluation, and user studies. In NAACL-ANLP 2000 Workshop on Automatic summarization