Project Report CS323: Text Summarization using Natural Language Processing

Group Members

1. Kanchumarthi H Pravallika - 200101049
2. Ashwitha Banoth - 200101021
3. Pruthvi Raj G N - 200101088
4. Gandrathi Srija - 200101030

Abstract

This document serves as a report for the course project in CS323 - Natural Language Processing. This project is based on extracting sentences. This report will highlight the challenges and insights in the problem statement. We drew inspiration from how humans summarize documents, sometimes combining and selecting multiple sentences for the summary. Moreover, we propose that the current algorithms (Frequency based algorithm, Luhn algorithm, Cosine similarity) are an adequate measure of performance for the summarization task. Text summarization provides framework to explicitly make a shorter version of one or more text documents. It is essential to extract the information in such a way that the content should be of user's interest. When method select sentences from word document and rank them on basis of their weight to generate summary then that method is called extractive summarization.

1. Introduction

Our main motivation for choosing this particular task was multifold. Firstly, text summarisation is a hard problem in NLP, due to the absence of standardized evaluation metrics, lack of *ground truth* summaries, and inability to manually annotate most training data to scale.

Extractive text summarization takes it a step further, adding the feature selection of sentences in documents. The summary produced not only has to be an accurate representation of the document, but also has to be succinct and grammatically and linguistically correct. The problem transforms to
"How would a human summarize this?"

To further add to this task, most of the current evaluation algorithms just compare the generated summary with the original document, making the assumption that this is the only acceptable summarization of this document, and anything different is considered bad. With this motivation, and an interest in extractive text summarisation, we had decided to choose the following as our project topic. We experimented with various datasets and looked at efficiencies of some of the algorithms to get some insight into our path for progress.

2. Methods

The task is split into a few sub tasks, namely selecting the sub-sentences to form the highlights, and to create a summary. So, we assume there exists an algorithm capable of selecting the apt sentences to form the summary. In a way, the algorithm chooses the salient sentences from the document and to form the summary.

Implement the following summarization algorithms step by step in Python: frequency-based, distance-based and the classic Luhn algorithm Use the following libraries for text summarization: sumy, pysummarization and BERT summarizer. Summarize articles extracted from web pages and feeds. We used the NLTK and spacy libraries and Google Colab for our natural language processing implementations. Created HTML visualizations for the presentation of the summaries.

1. Algorithms

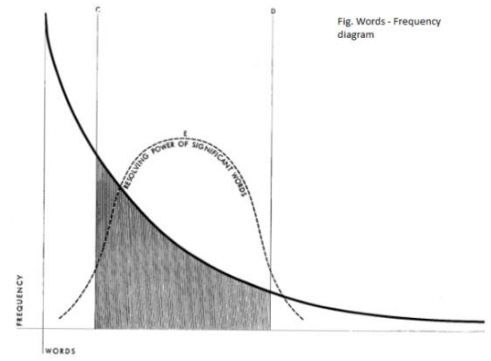
- Frequency based algorithm

Frequency based algorithm uses seven steps for generating the summary

1. Preprocessing the texts
   In this algorithm, upper case letters are changed to lower case letters to obtain same format, stop words are removed and punctuations are also removed.
2. Word frequency
   The frequency or the number of times the word is appeared after preprocessing the text.
3. Weighted word frequency
   Weighted word frequency for each word is calculated by taking the ratio of frequency of that word to the highest frequency obtained from the word frequency table.
4. Sentence tokenization
   Here, sentences are tokenized (splitting of sentences from the original text).
5. Score for the sentences
   Scores for the sentences are calculated by adding the weighted word frequencies for the words that are present in the tokenized sentences.
6. Order the sentences
   Sentences are ordered based on the scores.
7. Generate the summary
   Summary is generated by combining the top scored sentences.

- Luhn algorithm
  Luhn proposed that the significance of each word in a document signifies how important it is. The idea is that any sentence with maximum occurrences of the highest frequency words (stopwords) and least occurrences are not important to the meaning of document than others. Although it is not considered very accurate approach.

Luhn's algorithm is an approach based on TF-IDF. It selects only the words of higher importance as per their frequency. Higher weights are assigned to the words present at the beginning of the document.
In this method we select sentences with highest concentration of salient content terms. For calculating the significance instead of number of significant words by all words. It considers the words lying in the shaded region in this graph:



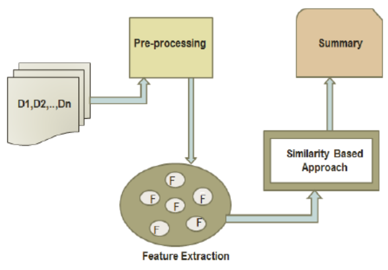
The region on the right signifies highest occurring elements while words on the left signifies least occurring elements. Luhn introduced the following criteria during text processing:

1. Removing stopwords
2. Stemming

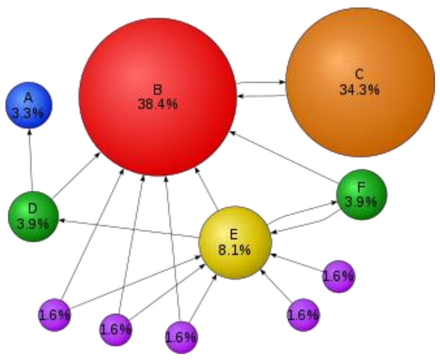
Luhns method is a simple technique in order to generate a summary from given words. The algorithm can be implemented in two stages. In the **first stage**, we try to determine which words are more significant towards the meaning of document. Luhn states that this is first done by doing a frequency analysis, then finding words which are significant, but not unimportant English words. In the **second phase**, we find out the most common words in the document, and then take a subset of those that are not these most common english words, but are still important.

- Cosine similarity

The cosine similarity falls under the extractive text summarization method. A measure of similarity between two non-zero vectors is cosine similarity. It can be used to identify similarities between sentences because we'll be representing our sentences as a collection of vectors. It calculates the angle between two vectors' cosine. If the sentences are comparable, the angle will be zero.



The cosine Similarity approach is as follows: Cosine Similarity measures the similarity between two sentences or documents in terms of the value within the range of [-1, 1] whichever you want to measure. That is the Cosine Similarity.
Relevant sentences are extracted and merged into one utilizing the cosine similarity approach after assessing the similarity-based approach and document relevancy. As a result, it generates a final summary after integrating the data.
We implemented PageRank algorithm, PageRank works by counting the number and quality of links to a page to determine a rough estimate of how important the website is. The underlying assumption is that more important websites are likely to receive more links from other websites.



3. Experiment

Dataset

All experimentation was done on the random Wikipedia text and non-anonymised version of the CNN/Dailymail dataset (modified for summarization). In frequency based algorithm we can give the number of sentences(top scored sentences)for generating the summary. In Luhn algorithm we can give the number of sentences and also the percentage of sentences such that optimised number of sentences are extracted from the original document that are required for generating the summary. In cosine similarity, score is calculated for each sentence by comparing its similarity with other sentences and the summary is extracted.

4. Conclusion & future scope

During the course of this project, we realized the need for developing an accurate evaluation metric for summaries which take into account the document itself (rather than some gold summary). This will lead to an evaluator which is able to understand the underlying structure of the document and summary, comparing these two, rather than just a word-by-word comparison as is the state of the art. But we believe by Human Evaluation we can showcase the improvements in our model. Each of these three algorithms have their own scoring methods for generating the summary. According to the results obtained from our experiment we conclude that by cosine similarity we can extract efficient summary compared with other two algorithms. In cosine similarity the higher similarity corresponds to the lower distances. When we pick the threshold for similarities for text or documents, usually a value higher than 0.5 shows strong similarities.

5. Acknowledgments

We thank Professor Dr. Ashish Anand for his support and guidance, as well as giving us this project opportunity in his course CS323 – Introduction to Natural Language Processing.