Keyword, Keysentence Extraction and Extractive Summarization

Nikhil Pinnaparaju
November 16, 2017

1 INTRODUCTION

Automatic summarization is the process of shortening a text document with software, in order to create a summary with the major points of the original document.

The two approaches to automatic summarization, Extractive Summarization and Abstractive Summarization.

Extractive methods work by selecting a subset of existing words, phrases, or sentences in the original text to form the summary, whereas, abstractive methods build an internal semantic representation and then use natural language generation techniques to create a summary that is closer to what a human might express. Such a summary might include verbal innovations.

This project goes over Extractive Summarization upon identifying the key words and key phrases and then selecting sentences from the passage based on scores obtained from previous stages.

The methods of implementation used involved building graph dependencies, term frequency - inverse document frequency values, etc. which will be covered in detail in the methods portion for the report. The data used for evaluation of the model is from the UCI machine learning datasets available online and mentioned in the references.

1.1 Stepwise Approach to Problem

- Extraction of Keywords
- Obtaining the Key Sentences
- Generation of Non-coherent summary
- Evaluation of the Model Built

2 Problem Statement

Problem Addressed:- *"How do we obtain the important aspects of a document* $given the document$ *using an algorithm?"*

3 Method Followed

There were various approaches taken towards this project and I will be covering all of them.

3.1 Textrank

The Primary approach I have taken to this problem, is the use of a Research paper by the *University of North Texas*, TextRank.
Edges are created based on word co-occurrence in this application of TextRank. Two vertices are connected by an edge if the unigrams appear within a window of size N in the original text. N is typically around 2 to 10. Thus, "natural" and "language" might be linked in a text about NLP. "Natural" and "processing" would also be linked because they would both appear in the same string of N words. These edges build on the notion of "text cohesion" and the idea that words that appear near each other are likely related in a meaningful way and "recommend" each other to the reader. Since this method simply ranks the individual vertices, we need a way to threshold or produce a limited number of keyphrases. The technique chosen is to set a count T to be a user-specified fraction of the total number of vertices in the graph. Then the top T vertices/unigrams are selected based on their stationary probabilities. A post- processing step is then applied to merge adjacent instances of these T unigrams. As a result, potentially more or less than T final keyphrases will be produced, but the number should be roughly proportional to the length of the original text.

3.2 TD-IDF Scores

In information retrieval, tfidf or TFIDF, short for term frequency inverse document frequency, is a numerical statistic that is intended to reflect how important a word is to a document in a collection or corpus. The tf-idf value increases proportionally to the number of times a word appears in the document, but is often offset by the frequency of the word in the corpus, which helps to adjust for the fact that some words appear more frequently in general. Nowadays, tf-idf is one of the most popular term-weighting schemes; 83% of text-based recommender systems in the domain of digital libraries use tf-idf.

4 Experimentation and Results

To test the system built, I wrote a python web scraper to scrape Wikipedia articles and then used it for basic testing and then manually checked the output. Although the manual method gave me a rough idea of the accuracy of the system I then used XML marked data, from the UCI machine learning dataset available online that comes with a full length passage along with its summary and then checked how my summary and the annotated summary compared, checking if all provided key phrases where present.
The results were fairly satisfactory and the system worked well on the dataset.

5 Discussion

5.1 Why it works

TextRank works because it does not only rely on the local context of a text unit (vertex), but rather it takes into account information recursively drawn from the entire text (graph). Through the graphs it builds on texts, TextRank identifies connections between various entities in a text, and implements the concept of recommendation. A text unit recommends other related text units, and the strength of the recommendation is recursively computed based on the importance of the units making the recommendation. For instance, in the keyphrase extraction application, co-occurring words recommend each other as important, and it is the common context that enables the identification of connections between words in text. In the process of identifying important sentences in a text, a sentence recommends another sentence that addresses similar concepts as being useful for the overall understanding of the text. The sentences that are highly recommended by other sentences in the text are likely to be more informative for the given text, and will be therefore given a higher score.

6 Conclusion

We can see from the accuracy of the system that it works fairly well on the given dataset. I am glad I was able to work on such a great topic and I learnt a lot from this experience.

7 References and Papers read

7.1

1. `https://web.eecs.umich.edu/~mihalcea/papers/mihalcea.emnlp04.pdf`
2. `https://homes.cs.washington.edu/~mausam/papers/naacl13.pdf`
3. `http://scikit-learn.org/stable/documentation.html`

4. https://archive.ics.uci.edu/ml/datasets/Legal+Case+Reports