INFORMATION RETRIEVAL

MULTI-DOCUMENT EXTRACTION BASED TEXT SUMMARIZER

Project Report

chandan Kumar Roy
S20180010036

Abstract

In this digital era, online contents have flooded the internet. People are busy in their own life and most of them keep themselves engaged with social media and smartphones. They better prefer watching video content than reading a paper. And most of the documents are multipage and lengthier. So, it is a good idea to come up with a text summarizer that can save people's time while reading and give them the desired amount of information within seconds.

Introduction

Natural language and Information retrieval have led plenty of researches in the domain of text summarizer. Most of the summarizer today uses sentence extraction on the basis of sentence importance score. Sentence scoring method uses both statistical and semantic feature to distinguish between important sentences and redundant sentences. Other kind of summarizer are abstractive. They rely upon abstract of sentences. Instead of inserting the original sentence, they modify the sentences semantically in the summary. Most common way to calculate the sentence importance score is to train a binary classifier, Markov model or directly assigning weight based on different similarity measures. In this report, I have implemented a multi-document extraction-based text-summarizer.

Similarity measures used in this project are –

- Cosine similarity
- Lesk sense Disambiguation (LSD)

Evaluation metrics used in this project is Recall Oriented Understudy for Gisting Evaluation (ROUGE) score.

Data Introduction

In this project, the dataset used for training purpose has been taken from **DUC2004** dataset. It consists of approximately 50 sets of 10 documents each.

Sentence Similarity Score

- **Cosine similarity score**
  It gives the cosine of angle between two vectors.it is represented as

$$sim = \frac{A \cdot B}{||A||||B||}$$

While deriving similarity between sentences, cosine similarity is not a good measure as it only looks for the similar word between the sentences. To calculate the semantic similarity between sentences, I have used Lesk algorithm.

- **Semantic Similarity**

  Semantic similarity between two sentences using Lesk algorithm is calculated as follows.

  1. Tokenization of sentences.
  2. Stemming.
  3. PoS (Parts of Speech) tagging.
  4. Find appropriate word sense using Lesk Sense Disambiguation (LSD).
  5. Find sentence similarity based on word pair similarity

- **Lesk Algorithm**

  The Lesk algorithm uses gloss to disambiguate polysemous word by counting number of words share among them. The more overlapping the words, more related are their senses.

  Once the more appropriate senses have been calculated for each word using Lesk algorithm, semantic similarity between the sentences are calculated using is-a (Hypernym-Hyponym) hierarchy in *WordNet*. We calculate similarity based on the path length. Shorter the path length, more similar are they. Similarity score of the entire sentence is the cumulative sum of individual similarity score of consisting words.

Sentence Importance Score

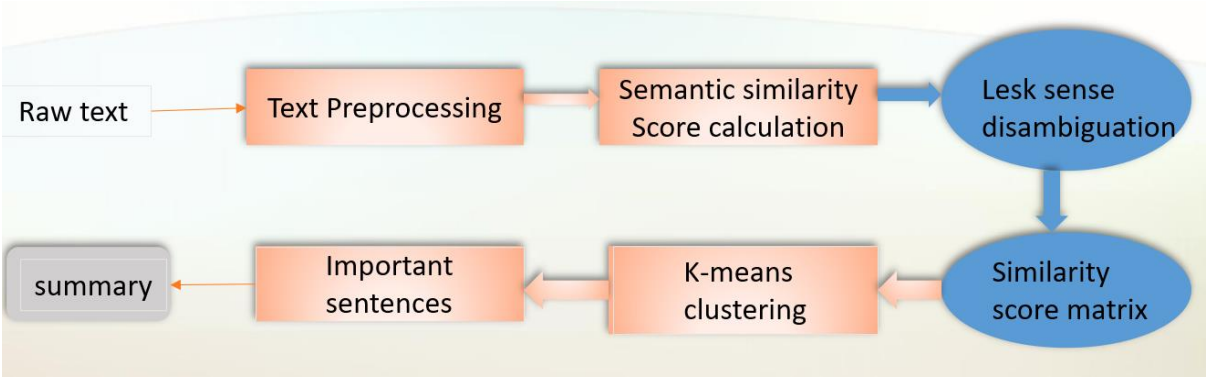
Each sentence in the text is given an importance score. Their importance in the text is proportional to their importance score. There are many ways to calculate sentence importance score in both supervised and unsupervised way. For supervised method we can train our model on the dataset based on features like Jaccard similarity score, tf-idf, word frequency etc... K-Means clustering can be used for unsupervised method. In this project, I have used Unsupervised way to calculate the sentence importance score for each sentence.

Algorithm

- ***K-Means Clustering***

  It is an unsupervised way of classifying the training set into categories. The procedure is to classify the training set into k number of clusteroids. It is an iterative process in which each of the k cluster is randomly initialized in the beginning. The next step is to take each data point and assign it to the nearest clusteroid. This is repeated until the clusteroids converges means there is no more shift in the clusteroid. Clusteroid of each of the cluster is added to the sentence summary. Distance metric used here is sentence similarity score. Sentences belonging to each cluster are nearly similar to their clusteroid in terms of similarity score.

Model Structure



I have performed unsupervised learning over **DUC2004** dataset. The model takes in directory path to raw text (in .txt format) and outputs the summary in around 100 words. It can be customized. Behind the scene, many tasks are performed as shown above. Text preprocessing involves text cleaning, stop word removal, stemming, normalization etc. It is followed by calculation of sentence similarity score using WordNet is-a hierarchy. Unsupervised clustering is performed on the similarity matrix using predefined number of cluster (k = 5). Clustering groups the sentences into 5 or a smaller number of clusters. Clusteroid of each of the cluster is extracted from the text and then added into the summary.

Performance on *DUC2004* Dataset

I have used ROUGE metrics to asses the performance of the model. I compared the model output with some baseline summary and it came out to be close to state-of-the art model. Average rouge-2 score over all the 50 documents are as below.

Average precision = 0.12

Average Recall = 0.06

Average f-score = 0.07

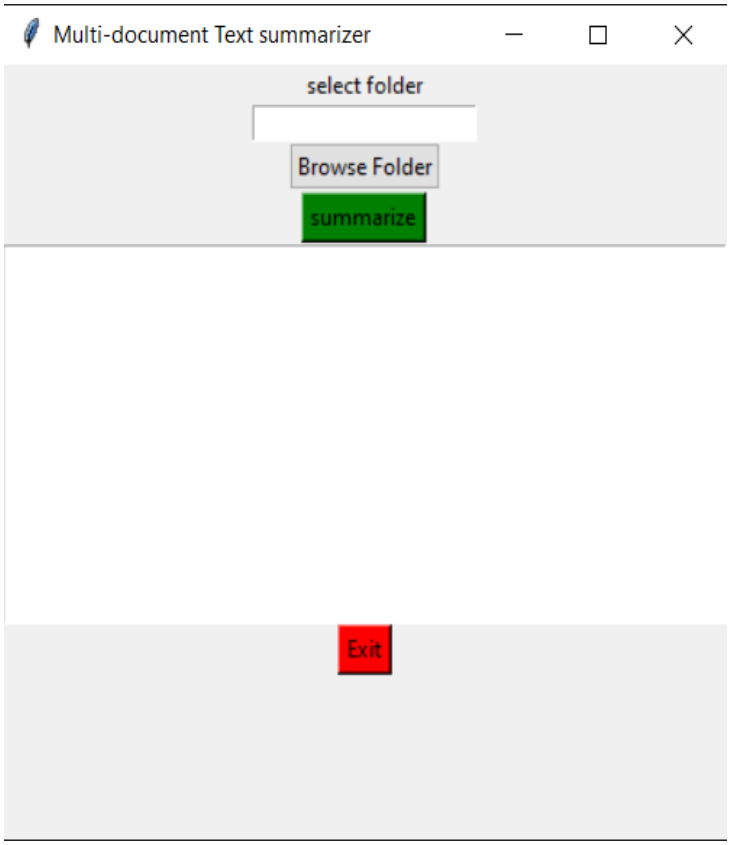
Future work

Many improvements can be done on the existing model. Instead of greedily selecting the importance sentences, bipartite graph based novel approach can be used. It will certainly improve the score. Neural-net based approach can be used like RNN and LSTM and It can significantly boost the model performance.

References

- **•https://nlp.stanford.edu/courses/cs224n/2010/reports/ssandeep-venuk-gkparai.pdf**
- **https://www-nlpir.nist.gov/projects/duc/data/2004_data.html**

Model UI



Demo