AUTOMATIC TEXT SUMMARIZATION

7th Semester 'Data Analytics' Final Project

Submitted by

Sushant Bhat (140911336, 42)

Sunny Bhalotia (140911462, 65)

DEPARTMENT OF INFORMATION AND COMMUNICATION ENGINEERING

MANIPAL INSTITUTE OF TECHNOLOGY

(A Constituent College of Manipal University)

MANIPAL – 576104, KARNATAKA, INDIA

Contents

INTRODUCTION

Abstract

A summary is a text that is produced from one or more texts, that contain a significant portion of information in the original text, and that is no longer than half the original text. Summarization is used extensively in generating search engine query results and in generating the automated abstracts of research papers. Summarization plays an important role in categorizing the ever-growing extensive collection of web pages that is present in the web today. The most difficult part for any summarizer is choosing which sentences are important and need to be selected for final summarization. Text summarizer has to be developed keeping in mind that there is no loss of information during summarization phase.

Problem definition

The goal of the project is to implement text summarization using map reduce programming model of Apache Hadoop framework, in order to present the most important information of the original text in a shorter version while keeping its main content unaltered and helping the user to quickly understand large volumes of information. Here we make use of extraction based summarization, where in the summarization task, the automatic system extracts objects from the entire collection, without modifying the objects themselves.

OBJECTIVES

The key objective of summarization would be to choose the most important sentences that would become part of the final summary. In order to achieve this we assign a numerical score to each sentence and rank the sentences accordingly. Calculating numerical score associated with a particular sentence depends on various criteria.

Term weight: The frequency of term occurrences within a document has often been used for calculating the importance of sentence. The score of a sentence can be calculated as the sum of the score of words in the sentence.

Sentence position: Text in the beginning of the document and towards the end of the document usually tend to provide much briefer insight about the context. Hence we would ideally want to preserve such sentences.

Sentence to Sentence Similarity: This feature is a similarity between sentences. For each sentence S, the similarity between S and each other sentence is computed by the cosine similarity measure with a resulting value between 0 and 1.

Thematic Word: The number of thematic word in sentence, this feature is important because terms that occur frequently in a document are probably related to topic.

Proper noun: Sentences that contain more proper nouns tends to be more important and it is most probably included in the document. Ratio of number of proper nouns that occur in the sentence to sentence length estimates the score for this feature.

Title feature: The word that occurs in sentence that also occurs in title gives higher score for the sentence. This is determined by counting the number of matches between words in sentence and words in the title.

IMPLEMENTATION

Preprocessing module

This module takes the plain text as input and produces refined text as output. Refined text is the resultant of the preprocessing step which doesn't include any stop words in it. Also some of the special characters like '(' are removed in order to ensure consistency across other modules. Result of this module is fed into execution module.

Execution module

Actual calculation of feature scores take place in this module which makes use of map reduce programming model Apache Hadoop framework. The refined text produced in the preprocessing module is read line by line by the mapper and it emits key value pairs of sentence, feature score. Reducer receives this sentence, feature score pairs and sums up all the feature score to determine the final score of the sentence.

Feature scores are calculated by making use of previously mentioned sentence ranking features. Computation of scores for these sentence ranking features take place in following way:

Term weight : twScore = tf*log(isf), where tf is term frequency of word, isf inverse sentence frequency.

Sentence positon: spScore = score/N, where N is number of sentences, score is defined as:

score = N – pos, if N – pos > pos

score = pos otherwise, where pos is sentence position.

Sentence to sentence similarity: ssScore = ∑cos(Scurr,Srest)/T, where cos calculates cosine similarity between current sentence and rest of the sentences, the summation of this is normalized by T, which is total words in the paragraph set.

Thematic words: tw = 0.1*n, where n is number of thematic words in the sentence. Word qualifies to be a thematic word if it occurs more times than a predefined threshold t.

Proper noun: pnScore = n/l, where n is number of proper nouns in the sentence and l is the length of the sentence.

Title score: tsScore = w/l, where w is the number of title word that sentence contains and l is length of the sentence.

Output produced by this module is <sentence, final score> pairs, which serves as input for the next module.

Extraction and pruning module:

So far the original sentences have got distorted and as it is they don’t make any sense. In extraction phase we take out the original sentence which corresponds with the preprocessed sentence. Once original sentences are retrieved it is pruned based on the final scores got in execution module to eliminate the lower scored sentences. Result of this module is written to a text file, which becomes our final summary.

CONCLUSION

Main idea of automatic text summarization is to find a subset of data which contains the "information" of the entire set. Such techniques are widely used in industry today. Whereas, MapReduce programming model achieves an associated implementation for processing and generating big data sets with a parallel, distributed algorithm on a cluster. When brought together these two techniques help us in interpreting much larger chunks of text in an efficient manner.

REFERENCES

[1] Ladda, Suanmali; Naomie, Salim and Mohammed, Salem *“Fuzzy Logic Based Method for Improving Text Summarization”* Bangkok, Thailand.