Automatic Text Summarization

AI assignment - 1

R Monith Sourya | 2016A7PS0006H

Nikhil Sreekumar Nair | 2016A7PS0049H

Sanjay Devprasad | 2016A7PS0033H

# Analysis Document

# Necessity and Motivation

With the onset of the internet revolution, there has been an explosion in the amount of information available for study. And in order to conduct studies aimed at appropriate and accurate conclusions, we require the latest information there is available. By building an automatic text summarizer, we can intelligently and accurately summarize documents to understand them better and shorten the time it takes us in achieving the same.

The main idea of summarization is to find a subset of data which contains the "information" of the entire set. Document summarization tries to create a representative summary or abstract of the entire document, by finding the most informative sentences.

# Existing systems and approaches

Typical text summarization makes use of either an *extractive* or an *abstractive* model.

Extractive methods work by selecting a subset of existing words, phrases, or sentences in the original text and concatenate them to form a summary. Extractive systems are quite robust since they use existing natural language phrases that are taken straight from the input. However, we must realize that they lack in flexibility since they cannot use novel words or connectors. They also cannot paraphrase like we, as humans, are capable of achieving.

In contrast, 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, while intended to keep the original content’s intent, can use words that were not in the original input. The primary goal of an abstractive model is to create more fluent and natural summaries. However, such a model constitutes a harder problem as we now require the model to generate coherent phrases and connectors.

Abstraction modelled summarization is still a very heavily researched field, which is why most established automatic text summarizers make use of extraction models. Clustering of phrases and sentences in extraction-based summarization make use of either a graph based, feature based, topic based, or grammar-based systems (explained a little more in detail in *project scope and feasibility*).

# Project Scope and Feasibility

We have elected to apply a combination of both the approaches in our version of this project. Considering that the prospects of the same are heavily researched topics in the present, we were unable to find literature that had established methods to achieve this. We have selected a few methods that we hope will work toward building a hybrid of the two approaches. All of these ideas will be implemented and tested for results in due course of time before the completion of the assignment, and the optimal combination will be selected.

## Extraction

#### Graph Base

A stochastic graph base model that computes the relative importance of text-units. It builds a graph from the document, with words as nodes and edges weighted according to relevance relations.

TextRank is the typical graph-based method and is based on the infamous PageRank system used by Google Search Engines. Works on the principle that the more the links to a textual-unit the better it is. The weighted edges are represented by an adjacency matric. This is then converted into a transition probability matrix by taking total probability of the links for each textual-unit. A ‘PageSurfer’ accordingly iterates this transition until convergence below a threshold.

TextRank regards words or sentences as pages on the PageRank. For the interests of this project, however, we will be using a LexRank. LexRank uses the sentence as node and the similarity as relation/weight (similarity is calculated by IDF-modified Cosine similarity).

#### Feature Base

The feature base model extracts the features of sentence, then evaluate its importance.

In any typical feature-based model, following features are used.

* Location of the sentence in the document
* Presence of various parts of speech like verbs
* Length
* Term frequency
* Proper Nouns
* Formatting and styles

A system is built to indicate references via pronouns to previous sentences and incorporated into the scoring mechanism.

A score is granted to each sentence accordingly. The sentences that are ranked with this score, tend to be repetitive in nature, often talking about the same subject material. Hence, it is important to not include new sentences that refer to the same information. We intend to achieve this in the abstracting layer of the project.

Another popular algorithm is the Luhn’s Algorithm which evaluates score based on frequency. However, we will not be using it owing to lack of multiple factors and a possibility that it wouldn’t work well with an abstraction layer.

#### Topic Base

The topic base model calculates the various topics of each document and then evaluates each sentence upon their reference to various topics and scores it accordingly. A priority order can be maintained in the topic list itself, thus paying more attention to sentences that refer to the important and recurrent themes of the document.

Latent Semantic Analysis (LSA) will be used to the detect the topic. It is an algebraic-statistical method that extracts hidden semantic structures of words and sentences. It employs the Singular Value Decomposition (SVD) algorithm to find the inter-relation between various sentences and words.

## Abstractive layer

While the extraction methods that we have described above will successfully provide important sentences that capture the essence of the document, they lack in terms of cohesive content and linguistic quality. We aim to employ two approaches in this regard: Compression and Sentence Fusion. Another alternative to be considered is Pointer-Generator Networks.

#### Sentence Compression

We intend to use Long Short-Term Memory (LSTM) Recurrent Neural Networks (RNNs) as published in a google research paper. It is a deletion-based sentence compressor where the task is to translate a sentence into a sequence of zeros and ones, corresponding to token deletion decisions. It outperforms typical trained models that have been given syntactic information (like PoS and NE tagging).

#### Sentence Fusion

Two problems that we face here are: the identification of phrases across sentences that convey common information and the combination of those phrases into a grammatical sentence. We elected to employ a general approach that was used by a multi-document summarizer called MultiGen and tweak its algorithms to suit the context.

Often similarities are identified using sentence clustering; each cluster is considered to represent a theme in the input. MultiGen uses a text-to-text generation technique, which takes as input a set of similar sentences and produces a new sentence containing information common to most of them.