An Automatic Document Summarization System

Jiadong Li
University of Washington
jiadongl@uw.edu

Abstract

Abstract-

keywords: documents, summarization, extraction.

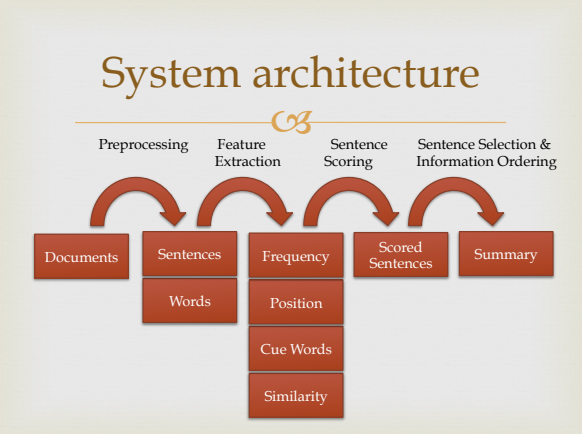
1 Introduction

In the paper, I will introduce an automatic system to generate summarization base on multiple documents about one same topic.

2 System Overview

The most important task in summarization is to extract the most relevant sentences of the topic among those documents. The overview approach is to use statistical and linguistic factors to identify the target sentences and use content realization technique to represent the summary well formatted. My Summarization system can be divided into 6 steps, Preprocessing, Feature Extraction, Sentence Scoring, Sentence Selection, Information Ordering, and Content Realization.

It can be illustrated as following diagram.



2.1 Preprocessing

The preprocessing step is to read in the topic documents and generate sentences list and words frequency dictionary. It contains the following 5 steps.

2.1.1 Sentence Segmentation

I use the nltk.sent_tokenize() method to implement the sentence segmentation. An additional process of striping the '\n', '\t' symbols is needed for the formatting purposes.

2.1.2 Word tokenization

I use the nltk.word_tokenize() method to implement the word tokenization. These words will contribute to forming a word count dictionary.

2.1.3 Stop word removal

Because the stop word and most punctuation are less important than other words, so those should be removed to prevent unnecessary influence.

2.1.4 Ignored word selection

When we build the word frequency dictionary, the nouns and verbs reflect the relatedness more than the determiner. So I build an ignored words list to filter out unrelated words.

2.1.5 Stemming

A word can be found in different forms, but they all represent the same meaning, so the stemming is really needed.

2.2 Feature Extraction

After the preprocessing work is done, we can extract four importance features from the previous data. They are Frequency, Sentence position, Cue words, and Similarity with the title.

2.2.1 Frequency

Frequency score is calculated by the word frequency dictionary. If a word's frequency is high, that means the word has a very important effect on the content of the topic. The sentence frequency score is calculated by sum up the frequency of every word in the sentence. In order to solve the long sentence issue, I introduced an average frequency score by divide the total frequency score by the count of the sentence.

2.2.2 Sentence position

Position of a sentence in a paragraph in a document has a meaning of the purpose of the sentence. By giving different weight based on the position, we can generate a position score.

2.2.3 Cue words

Cue words are connective expressions that links spans of communication and signals semantic relations in a text.

2.2.4 Similarity with the title.

The similarity score is calculated by the similarity between topic title with description and the sentence.

2.3 Sentence Scoring

In the (Sarraf and Meena 2014), they have an approach to build a linear combination of the frequency, sentence positional value, weights of cue words and similarity with the title of the documents, but I have built a different approach to calculate the score. My implementation is let SentenceScore = (FrequencyScore + SimilarityScore) * PositionScore * (1 + (CueWordScore * 0.1)).

2.4 Sentence Selection

When we get the scored sentences, we will rank them by the score decreasingly. The top N sentences will be selected for the summary until the total words count reaches 100.

2.5 Information Ordering

The selected sentence will be ordered by the sentence score decreasingly. If two sentences' score are kind of similar, then the order of doc will influence of the ordering. We don't care about the sentence position inside of the doc.

2.6 Content Realization

The content realization is not implemented yet in my D2.

3 Approach

3.1 Long sentence vs. short sentences

There is an issue about my system that it prefers the long sentence rather the short sentence. I have made a change to replace the frequency score of the sentence by the average frequency of the sentence.

4 Results

We are using ROUGE scores (Radev, et.al., 2004) to evaluate our system result. Here are the scores based on the data from: training/2009/UpdateSumm09_test_topics.xml

	P	R	F
ROUGE-1	0.31810	0.23876	0.27102
ROUGE-2	0.10376	0.07747	0.08814
ROUGE-3	0.03985	0.02973	0.03386
ROUGE-4	0.01939	0.01464	0.01660

5 Discussion

The ROUGE scores show that our system still have some room to improve.

1. We can use tf•idf method to improve similarity scoring.
2. I am still tuning the sentence score formula, maybe I can use the some training to get all the best parameters in order to have the best performance.
3. We can implement the content realization to get a better summary.

6 Conclusion

This paper discusses an approach to build an automated documents summarization system.

Reference

Dragomir R. Radev, Hongyan Jing, Malgorzata Stys, & Daniel Tam. 2004. *Centroid-based Summarization of Multiple Documents*. Information Processing and Management 40. Pages 919-938.

Daniel Jurafsky and James H. Martin. 2008. Speech and Language Processing. Prentice Hall, Upper Saddle River, NJ.

P. Sarraf, Y. K. Meena, Summarization of Document using Java ,Vol. 3, Issue 2, February – 2014 International Journal of Engineering Research & Technology(IJERT).