An Automatic Document Summarization System

Jiadong Li
University of Washington
jiadongl@uw.edu

Abstract

Topic-oriented documents summarization is a process of creating a shorter version of multiple topic-related text documents. It is a very important way of finding relevant information in large text pool. The text extraction techniques select entire sentences from documents according to some criteria to form a summary. The mostly used technique is the sentence scoring. This paper advocates a new method of calculating the sentence score and generating the summary.

Keywords: documents, summarization, extraction.

1 Introduction

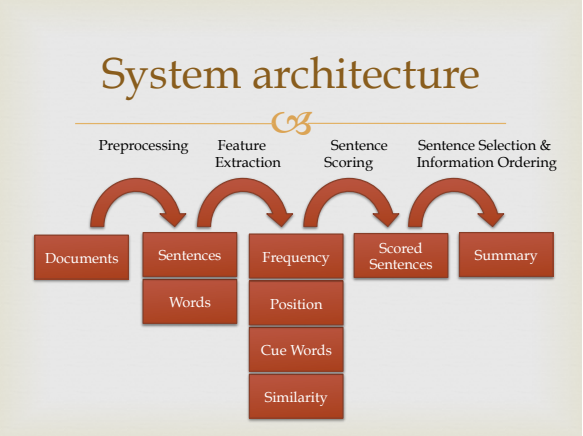
In the paper, I will introduce an automatic system to generate summarization base on multiple documents about one same topic.

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.

(This diagram is the initial design, the major steps are not changed, but the detail steps may have some difference in the D3&D4, will update it in D4.I will also add the last step of Content realization.)



3 Approach

3.1 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.

3.1.1 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.

3.1.2 Word tokenization

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

3.1.3 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.

3.1.4 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.

3.1.5 Stemming

A word can be found in different forms, but they all represent the same meaning, so the stemming is really needed. After some comparison, we selected the snowball stemmer. You can call nltk.stem.SnowballStemmer("english") to use it. It provides a much better performance than the default PorterStemmer.

3.2 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.

3.2.1 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.

After the first trial, we find the sentence selection result dose not meet our expectation, then we realized that issue does not lie on the length of the sentence. A long sentence should be favorable because it carries more information for summary. The real problem is that we should filter out those long sentences with too many nonsense words, which mean the frequency score for those words is high, but the information contribution is low.

In order to solve that problem, we have changed the way we build our word frequency dictionary. At first, we use nltk.pos_tag(words) to give every words a part of speech tag. It includes all the part-of-speech tags used in the Penn Treebank. Here is a list of them.

 # N Tag Description
 # 1. CC Coordinating conjunction
 # 2. CD Cardinal number
 # 3. DT Determiner
 # 4. EX Existential there
 # 5. FW Foreign word
 # 6. IN Preposition or subordinating conjunction
 # 7. JJ Adjective
 # 8. JJR Adjective, comparative
 # 9. JJS Adjective, superlative
 # 10. LS List item marker
 # 11. MD Modal
 # 12. NN Noun, singular or mass
 # 13. NNS Noun, plural
 # 14. NNP Proper noun, singular
 # 15. NNPS Proper noun, plural
 # 16. PDT Predeterminer
 # 17. POS Possessive ending
 # 18. PRP Personal pronoun
 # 19. PRP$ Possessive pronoun
 # 20. RB Adverb
 # 21. RBR Adverb, comparative
 # 22. RBS Adverb, superlative
 # 23. RP Particle
 # 24. SYM Symbol
 # 25. TO to
 # 26. UH Interjection
 # 27. VB Verb, base form
 # 28. VBD Verb, past tense
 # 29. VBG Verb, gerund or present participle
 # 30. VBN Verb, past participle
 # 31. VBP Verb, non-3rd person singular present
 # 32. VBZ Verb, 3rd person singular present
 # 33. WDT Wh-determiner
 # 34. WP Wh-pronoun
 # 35. WP$ Possessive wh-pronoun
 # 36. WRB Wh-adverb

We have set a POS_tag_weight dictionary: "CD": 1, "JJ": 1, "JJR": 1, "JJS": 1, "NN": 5, "NNS": 5, "NNP": 5, "NNPS": 5, "PRP": 1, "PRP$": 1, "VB": 5, "VBD": 5, "VBG": 5, "VBN": 5, "VBP": 5, "VBZ": 5. It gives the Noun and Verb weight 5, the Adjective words a weight 1 and ignore other tag of words, since we understand that the noun or verb plays a much more important role in the sentence meaning. By doing this, we build a new word dictionary and calculate the frequency score of the whole sentence without normalization. And we don't need to worry about the long sentence issue because the nonsense words are all ignored.

3.2.2 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. The basic approach is like:

The baseline score for each sentence is 1.

If it is the first paragraph, the position score will be multiplied by 1.2.

If it is the first sentence in the paragraph, the position score will be multiplied by 1.2.

That mean the first sentence of the document will be weighted 1.44.

3.2.3 Cue words

Cue words are connective expressions that links spans of communication and signals semantic relations in a text. This is a very interesting idea introduced in (Sarraf 2014), but in my practice, it seems not that useful. Because the cue words are very rarely appeared, and it contributes very little to the clue of target sentences, so in my updated approach, I end up removing this feature.

3.2.4 Similarity with the title.

The similarity score is calculated by the similarity between topic title with description and the sentence. In my first approach, I was using the TAC-2009 topic-oriented document sets. All the documents have tile names and descriptions and the topic has narratives, so I used a very complex method to compare and calculate the similarity between the documents and topic. When I switched my document sets to the TAC-2009, I found not all the documents have descriptions and not all the topics have narratives. My first approach is not working at this time.

In my updated approach, I have created a simplified method to calculate the similarity score. It is to tokenize and stem the topic words, and give them the weight as maximum frequency count weight. That means we will treat the topic words as the same weighted clues as the most frequent word in that topic to compare the similarity for each sentences with the topic.

3.3 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)).

In my updated approach, as the cue word feature is cancelled, so the sentence score formula changes into (FrequencyScore + SimilarityScore) * PositionScore.

3.4 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.

3.5 Information Ordering

The selected sentence will be ordered by the chronological expert. If two sentences are from different document, they will be ordered by document timestamp. If two sentences are from same document, they will be ordered by order within document. To implement this, I have created a position index tag in each sentence like [doc_index, paragraph_index, line_index]. We can directly sort the sentences by this position tag.

3.6 Content Realization

The content realization is not implemented yet in my D3.

4 Results

We are using ROUGE scores (Radev, et.al., 2004) to evaluate our system result.

For Deliverables 2 and 3, we used the TAC-2010 topic-oriented document sets and their corresponding model summaries as my test data.

Here are the ROUGE score for my second deliverable:

	R(%)	P(%)	F(%)
ROUGE-1	17.026	24.534	19.931
ROUGE-2	4.88	7.093	5.73
ROUGE-3	1.762	2.639	2.092
ROUGE-4	0.655	1.011	0.788

And this the ROUGE score for the third deliverable:

	R(%)	P(%)	F(%)
ROUGE-1	22.887	28.247	25.085
ROUGE-2	6.366	7.789	6.954
ROUGE-3	2.158	2.658	2.363
ROUGE-4	0.895	1.115	0.985

As you can see, my updated approach has improve the average recall score of ROUGE-1 by almost 6%. I will give the credits to the new approach of forming the word frequency dictionary and chronological ordering. We believe that after we implement the content realization part, the ROUGE score will continue to increase.

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.
4. In the POS tag weight part, we have another thought to improve the system. In the current system, we give PRP(Personal pronoun) and PRP$(Possessive pronoun) a weight of 1. But actually, we should had a better way to do this. Since most of the PRP and PRP$ are representing a proper

noun or regular noun. We should firstly map them to the correct noun and add it to the frequency dictionary, because simply counting the PPR's frequency provides nothing-meaningful information to the sentence.

5. We are also working on a question about how to fully utilize the 100 words. There are couple thoughts now. The first one is always selecting one more sentence after reach 100 words and rewrite the summary into 100 words by removing JJ or PP in the content realization step; the second approach is to use document clustering, select one top sentence for each cluster.

6 Conclusion

This paper discusses an approach to build an automated document summarization system. It involves six major steps of Preprocessing, Feature extraction, Sentence Scoring, Sentence Selection, Information Ordering and Content Realization. We also discussed several implementation key point and we used the ROUGE score to evaluate our summarization system. Based on the ROUGE score, we can see that our system's score keeps growing for each deliverable change, and it provides a relative good performance.

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).