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. The quality of the generated summary will be measured by human readability and the Recall-Oriented Understudy for Gisting Evaluation (ROUGE).

Keywords: documents, summarization, natural language processing, sentence scoring, rouge score.

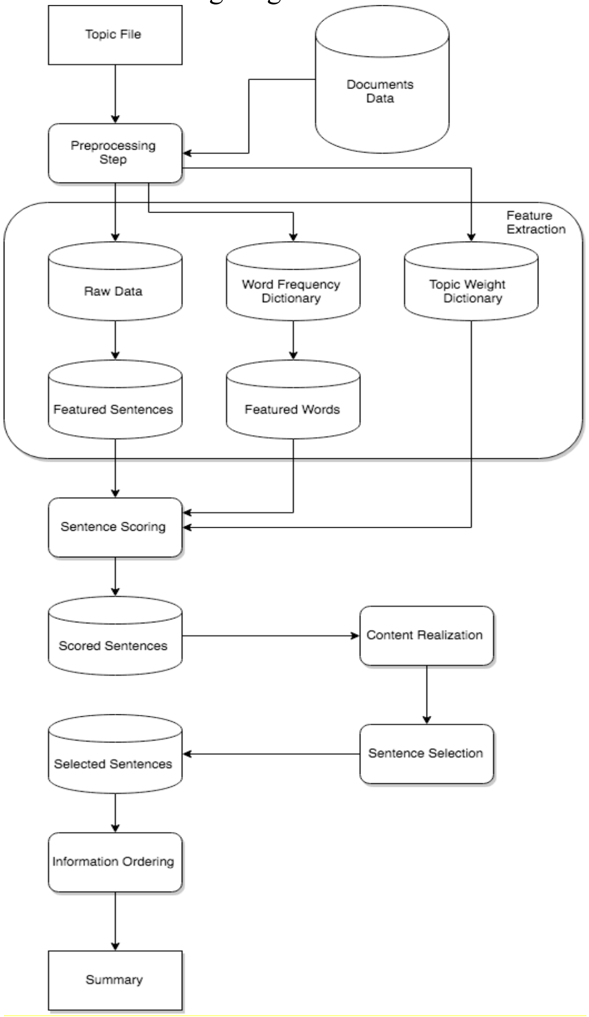
1 Introduction

There is more and more information around the world; we got less and less time to consume all the content. By the development of natural language processing technique, automatic document summarization system becomes a good solution to extract the most important information among large information pool in a short summary. With the help form my professor and shared experience from my classmates, I have tried a new approach to solve this problem and gotten some relatively good results. In the paper, I will introduce this new automatic system to generate summarization base on multiple documents about one same topic, and I will analyze the result and provide some further thought in the future development.

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, Content Realization, and Information Ordering.

The detailed system structure can be illustrated as following diagram.



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 20 sentences will be selected in a pool and prepared for the final summary generation. Since the project requirement from the course limits our summary word count to be less than 100, so we have to figure out a way to squeeze more and most informative information in the 100 words. For each pre-selected sentence, we will use the content realization technique to rewrite the sentence. Normally, the newly generated sentence is shorter than the original sentence. I will loop through the pool from the highest score to the lowest score and keep add qualified sentence into the final summary pool. I have three key factors to evaluate the qualification of the generated sentence. Firstly, the sentence's words count should not exceed 100. Secondly, after adding this sentence, the whole summary's words count should not exceed 100; Lastly, this sentence should not be the redundant sentence.

According the previous researches, there are many ways to detect and identify sentence redundancy. For example, one approach is to use term frequency–inverse document frequency (tf-idf) weighting to calculate the similarity between two sentences. Another approach is to build the k-means document cluster and avoid picking sentences from the same cluster. In my system, I selected to use word co-appearance ratio considering of the remaining time and implementation difficulties. This method is to loop through all previous qualified sentence and look for the count of same words, and calculate the ratio by diving the count to the current sentence's total words count. If the ratio exceeds 60%, then it will be recognized as redundant sentence and will be skip for selection.

3.5 Content Realization

For the content realization, I was using UMD Approach introduced in (David 2007) to implement sentence compression. The adjusted steps to trimmer the sentences is following:

a) Split the sentence by any punctuation.

b) Remove shorter phrase (<=2) at the beginning. This is meant to remove temporal expressions and news source.

c) Remove phrase in between two '--'.

d) Remove unnecessary punctuation.

e) Remove complementizer that.

f) Remove complement phrase between two punctuations if it is after a NN tag and before a VB/W tag.

g) Remove some determiners before Capitalized Proper Noun.

h) Remove some adv.

i) Remove PPs that do not contain Nes.

After the trimmer, I also need to strip all unnecessary space among sentence to make the summary more grammatical and clean.

In addition to the sentence compression, I have also investigated another way to solve redundant entity issue. The main approach is to use the Stanford Named Entity Recognizer (NER) to identify redundant entity, and use a supervised trained classifier model to replace them with proper pronoun. The approach has been examined by several groups of my classmates, but I just don't have the enough time to make it working for my system.

3.6 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.

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.

After we implemented the content realization, I released the fourth deliverable. Unlike the expectation of keeping increased ROUGE score, the actual result stays almost the same with some improvement on the recall score of ROUGE-1 by 2% and slight drop on all the other score. Here is the ROUGE score for the fourth deliverable:

	R(%)	P(%)	F(%)
ROUGE-1	24.605	26.078	25.156
ROUGE-2	6.611	6.936	6.751
ROUGE-3	2.072	2.159	2.108
ROUGE-4	0.786	0.812	0.796

Even though the ROUGE score doesn't show much improvement of the system, but when we compare the summary output from deliverable 3 and deliverable 4 side by side, we can easily find the improvement of readability and reduction of content redundancy. Even we fail to increase the ROUGE score through our deliverable 4, but the implementation of content realization contributes a lot on the readability and quality of the final summary. It is still very valuable to have it in the system.

In the end, I have applied the system to run it on the TAC-2011 topic-oriented document sets and their corresponding model summaries as the evaluation data set. Here is the ROUGE score for my final system on the evaluation data set:

	R(%)	P(%)	F(%)
ROUGE-1	27.908	29.233	28.463
ROUGE-2	7.452	7.783	7.593
ROUGE-3	2.445	2.532	2.482
ROUGE-4	1.039	1.056	1.045

As you can see, the final evaluation result is surprisingly much better than the development result. It is a good sign that we have not over fit our system.

5 Discussion

5.1 Error Analysis

5.1.1 Feature Selection

Comparing the result of D2 and D3, we can find the performance has a huge jump. The major contributor is the new approach of feature selection and new design for word frequency dictionary. The new approach abandoned unusual features like cue-words and import POS tagging to give different words a different weight, which emphasized the nouns and verbs.

5.1.2 Sentence score formula

During the development, the sentence score formula has been experimented several times with different combination. The current formula is already far different from its original form in (Sarraf and Meena 2014), but the performance is much better that the old one. There is still improvement space by adjusting the parameters in current formula. One approach is to use machine-learning algorithm to build a regression model to calculate the best-performance parameters. I failed to do that in the terms of time.

5.1.3 Sentence compression

My selection of sentence compression is rule based, so it is unavoidable to have situations that certain newly generated sentence is not grammatical or the readability is reduced. But when you look at the whole system, the general performance is increasing, so it is acceptable to sacrifice some minor sentences.

5.1.4 Redundant Entity

This is not implemented in my system, even though it will not affect the ROUGE score very much, but I have to admitted that the readability in this part is obviously not as good as other teams' result.

5.2 Future Work

With the dedicated work in the short period of time, it is already satisfied enough for me to have the system performing in this level. But there is still some future work that can be plan if you get the enough time. The first one is to implement the k-means document clustering at the feature extraction step, which can provide valuable information when facing the content selection. It is predictable that when we select each top sentence form each cluster as pre-selected sentences, the final summary will have a better coverage while keeping low redundancy. Another plan is to continue working on the entity recognition and redundant replacement approach to improve the readability.

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).

David Zajic, Bonnie J. Dorr, Jimmy Lin, Richard Schwartz. 2007. Multi-candidate reduction: Sentence compression as a tool for document summarization tasks, Information Processing & Management, Volume 43, Issue 6, Pages 1549-1570, ISSN 0306-4573.