The CIST Summarization System at TAC 2010

Hongyan Liu, Qing Zhao, Ying Xiong, Lei Li, Caixia Yuan

Center for Intelligence Science and Technology

Beijing University of Posts and Telecommunications

Beijing, 100876, China

Abstract

This is the first time we participate in TAC. In this report, we present our extractive summarization system on both initial and update summarization tracks of TAC 2010. We introduce an integrated method to generate all summaries. The TAC evaluation of results show that our summarization method is feasible but it has to be improved in future.

Keywords: multi-document summarization; Hierarchical clustering; sentence extraction; sentence compression; update summary

1. Introduction

With the fast development of the Internet and the emergence of massive amounts of information, we need the ability to find important semantic content quickly. But with the accelerated pace of life and the explosive increase in the number of documents, people who want to learn about a particular event have no time to read all the available documents on that topic. We need an effective way to generate a summary that lets people acquire important topic information. Multi-document summarization aims to generate a brief and coherent summary, which should be objective and exactly reflect the contents of the original documents and minimize redundancy[1]. Readers can obtain the important major topic content that they need without reading the entire original document set. At the same time, people want to know the progress of the topic they are interested in, which is what motivates the research behind update summarization. In this report, we also present a method to produce update summaries.

In this report, we propose an integrated extractive multi-document summarization framework based on sentence level semantic analysis. These summaries will be evaluated for readability and content and overall responsiveness. The remainder of this report is organized as follows: Section 2 introduces the related work. Section 3 discusses the framework of our method. Section 4 describes the details of generating original summaries. Section 5 introduces sentence compression and final summary generation. Section 6 presents the process of generating update summaries. In section 7 we analyze TAC evaluation results. Section 8 concludes the report with directions for future research.

2. Related Work

Multi-document summarization has been widely studied in recent years. In general, there are two types of methods: extractive summarization and abstractive one[2] [3]. The latter is more complicated since it involves language generation, information fusion and more natural language processing (NLP) technologies. Most work today focuses on extractive summarization, where a summary is created simply by identifying and subsequently concatenating the most important sentences in document set. Extractive summarization usually ranks the sentences in the documents according to their scores calculated by a set of predefined features, such as term frequency-inverse sentence frequency (TF-ISF)[4][5], sentence or term position[5], and number of keywords. Other methods include NMF-based topic specific summarization[6], CRF-based summarization[7], and hidden Markov model (HMM) based method[8]. In addition, some graph-ranking based methods are also proposed[9]. Most of these methods ignore the dependency of semantics in the sentence level and just focus on keyword co-occurrence. The hidden relationships between sentences need to be further discovered.

3. Framework

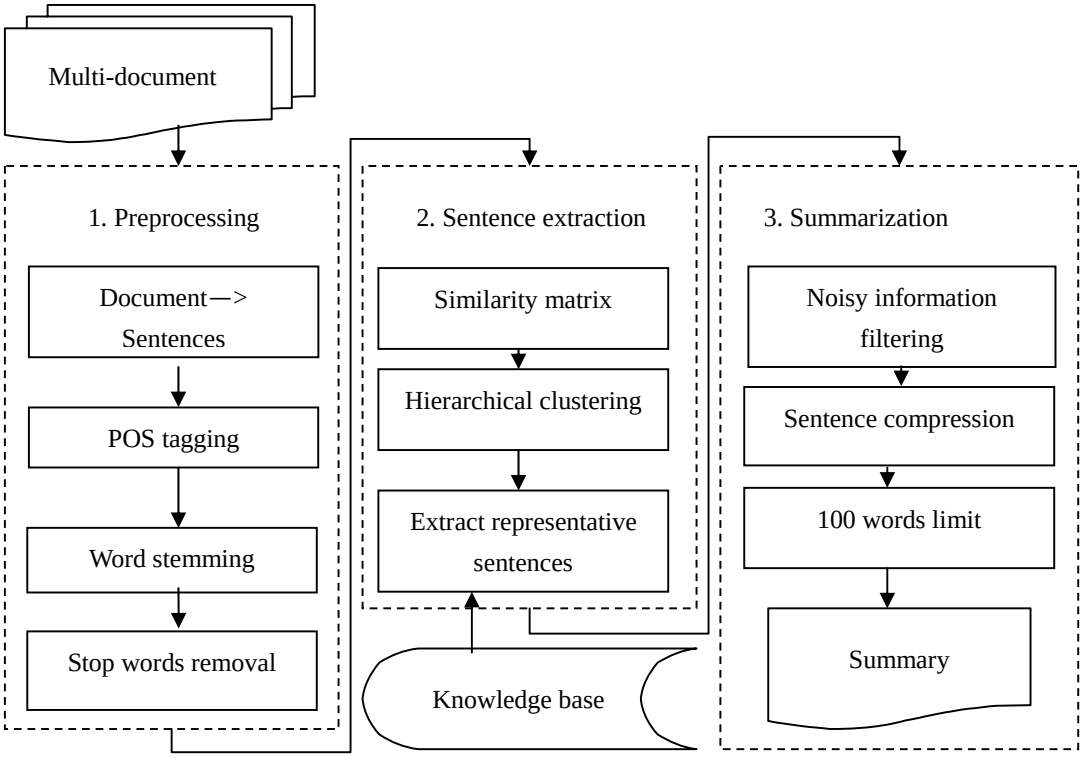


Figure 1 Multi-document Summarization Based on Clustering

Figure 1 shows the three key steps of our summarization system, which are preprocessing, sentence extraction and summarization. In the following parts we will introduce the details.

4.Generating Original Multi-document Summary

4.1 Preprocessing

The original documents that TAC provided as test data must be pretreated, including content extraction, POS tagging, word stemming and stop words removal.

Firstly, the original document set contains some tags, such as “<DOC>”, “<DOCNO>”, “</DOCNO>”, “<DOCTYPE>”, “</DOCTYPE>”, “<HEADER>”, “</HEADER>”, “<BODY>”, “<SLUG>”, “</SLUG>”. We delete them and only extract the paragraphs between and , <text> and </text> for summarization.

The final summary should be very short and informative. We choose the level of sentence. We segment the original English document into sequence of sentences with BFSU English Sentence Segmenter (http://www.corpus4u.org).

As is well known, nouns and verbs are dominant words expressing the content. We give them higher weights than the others. We perform POS tagging with an English POS tagger of Tokyo University. It can help us make a deeper semantic analysis of the source documents.

As English words are different from Chinese, people often use different word forms to represent the time and state of event, word stemming is necessary. We compute similarity of sentences after word stemming, which is supposed to be more precise. It helps us improve the clustering result. We perform word stemming with an English tool downloaded from “http://www.12fanyi.cn/post/83.html”.

The last step of preprocessing is stop words removing. In English documents we may find many words, such as “a”, “an”, “and”, “the”, “of”. They can’t directly express the content, only play a supporting role. We use a list of more than 300 stop words. After the removal of stop words, we get content words that contribute to deeper semantic

analysis.

4.2 Sentence Extraction

After preprocessing, the most critical step of producing multi-document summary is candidate sentence extraction.

(1) Sentence Similarity Matrix

Let sentence s_1's feature vector be (W_1 W_2 ..., W_r), r is the number of features, W_i is the weight of each feature,

$$Wi = TFISFi * POSi \quad (4\text{-}1)$$

$$TFISFi = tfi * \log N / sf \quad (4\text{-}2)$$

tf_i is the frequency of the feature in the sentence, sf is the number of sentences in which the feature appears, N is the total number of sentences. POS_i is the part of speech information. By doing so, we can make similarity calculation more accurate.

Let sentence s_2's feature vector be (W_1 W_2 ..., W_l). The similarity between s_1 and s_2 is calculated as follows:

$$Sim(s_1, s_2) = \frac{\sum w_x}{\sum w_i} \quad (4\text{-}3)$$

$$w_x \in s_1 \cap s_2, w_i \in s_{\min}$$

Numerator is the sum weight of the words that both occur in sentence s_1 and s_2. Denominator is the sum weight of the words that in the shorter sentence s_{min} in {s_1, s_2}. The benefit is that if a sentence contains all the words of another sentence, i.e. if one sentence is totally a part of another, then their similarity is 1. This is reasonable in this task, since in the next step of clustering, it will make the two sentences definitely belong to the same cluster.

(2) Hierarchical Clustering

Clustering based on sentence level can put sentences with related meaning together. Then we select representative sentences of each cluster to compose the summary. In our system we choose the hierarchical clustering algorithm.

The specific method is described below:

(1) Calculate the similarity of every two sentences, construct an m*m similarity matrix, where m is the number of sentences in the document set.

(2) For (when the update largest value in the matrix is less than the given threshold)

(2-1) Group the two sentences corresponding to the largest value.

(2-2) Update the similarity value between the remaining sentences and the newly grouped sentence. Suppose that we combine sentence s_a and s_b, the similarity between sentence m and the newly merged sentence is:

$$sim(s_m, s_{a\&b}) = \max\{sim(s_m, s_a), sim(s_m, s_b)\} \quad (4\text{-}4)$$

The benefit of hierarchical clustering [14] is that we needn't to decide the number of clusters. All we have to do is to decide the threshold, above which we consider the two sentences should be merged together.

4.3 Sentence Scoring

Now we have got many different clusters. The following task is to extract candidate sentences from each cluster. We expect that we can choose the most representative and non-redundant sentences. We evaluate sentences mainly according to keywords coverage and sentence length.

Firstly, we build a knowledge base for the required aspects listed in the guided summarizations. We extract keywords from TAC sample document sets for all the five topics which contain tagged information units according to the required aspects. We obtain an original version of the knowledge base for each topic with many required aspects and each aspect corresponds to a keyword list. Then we expand these keywords with thesaurus of Britannica Online Encyclopedia (http://www.britannica.com/bps/thesaurus?query=good) using a simple meta search engine. To ensure the quality of synonyms, in the final version of the knowledge base, we improve it through further manual screening. We expect that the knowledge base will help to select sentences which tightly cover the required aspects.

In order to emphasize those required aspects, we set higher weight for sentences containing more keywords in the knowledge base.

$$s_{asp} = \sum_{i=1}^{m} \sum_{j=1}^{S_l} w_j \quad (4\text{-}5)$$

The s_{asp} is the score of the sentence aspect coverage, and w_j is the score of each word in the sentence, if the word is in the knowledge base, the value is 1, else is 0, s_l is the total number of the words in the sentence, m is the total number of aspects of the category.

Then we mainly think of the other two features to decide the importance of the sentence:

1. the length of the sentence;
2. the number of keywords the sentence contains

In our experiment, Len(sen) is a Boolean function, if the length of one sentence is between 5-20 words, Len(sen)=1, otherwise Len (sen)=0; we

select keywords based on score of the words by formula (4-1) (4-2) and Hypothesis testing [14].

We select top 10 words as the keywords, and set

$$s_k = num(keywords \cap words(sen)) \quad (4\text{-}6)$$

s_k is the number of the keywords that a sentence contain, words(sen) are the words in sentence. We sort the initially chosen sentences in accordance with descending order of score computed by formula (4-7),

$$s = Len(sen) * s_k * s_{asp} \quad (4\text{-}7)$$

The s is the final score of the sentence, s_{asp} is calculated by formula (4-5).

Then we implement an optimal sentence selection in two stages.

Stage1: select sentences meeting certain conditions;

Stage2: delete the sentence carrying the least information until the remaining sentences meet with our goal.

The two stages are described more detailed in[10]. The algorithm is the overall optimization compared with the traditional methods that choose the currently best sentence in the unknowing of following-up sentences.

5. Final Summary

Since the final summary must meet some additional requirements such as overall length and grammaticality etc., we should remove the redundant information from the sentences. Sentence compression is often regarded as a promising method towards ameliorating some of the problems associated with extractive summarization. It involves creating a short grammatical summary of a single sentence, by removing elements that are considered extraneous, while retaining the most important information [11]. Interfacing extractive summarization with a sentence compression module could improve the conciseness of the generated summaries and render them more informative [12].

We mainly use Stanford Parser (http://nlp.stanford.edu:8080/parser/) to parse sentences, then select and combine phrases grammatically which tell the main concept of the sentence. Since some noisy information may interfere the parsing results, we remove them at the very beginning; this will be introduced in section 5.1. A detailed description will be given in section 5.2. Section 5.3 introduces some additional operations to modify the summary so as to satisfy all the constraints.

5.1 Noisy Information Filtering

There are a lot of useless punctuation marks (*, ``, ' ', etc.) and symbols (*&QL*, etc.) in the original summary generated in section 4. They not only cause the text length exceeding limit, but also, worse still, they often lead to errors in sentence parsing.

Take the sentence in file "D1020D-A" for example:

&UR; (Charles Downey is a Big Bear City, Calif.-based free-lance writer who frequently writes on parenting issues.) &LR; &UR; ---------- &QC; &UR; (To purchase this article, contact one of these New York Times Syndicate sales representatives: (_ U.S., Canada and the Pacific: CONNIE WHITE in Kansas City at 800-444-0267 or 816-822-8448; fax, 816-822-1444.

Firstly, it has fifty-three words, more than a half of the total word limit. Characters like "*&LR; &UR; ---------- &QC; &UR;*" contribute nothing to the keynote of the sentence. We need to delete all these noisy information.

Secondly, errors may occur when extracting keywords from the collapsed dependencies. For example, parts of the collapsed typed dependencies of this sentence are as followed:

nn(Charles-3, &UR;-1)
nsubj(City-9, Charles-3)
...
nsubj(writes-16, writer-13)
...
nsubj(----------3, &UR;-2)
nn(&UR;-5, &QC;-4)
nsubj(purchase-8, &UR;-5)
aux(purchase-8, To-7)
xcomp(---------- 3, purchase-8)
det(article-10, this-9)
dobj(purchase-8, article-10)
...

We use "nsubj" as the symbol to mark a sentence's subject-predicate, but from above we can see that there is no keyword in line 4.

If we delete the noisy words, the new sentence becomes "*Charles Downey is a Big Bear City, Calif.-based free-lance writer who frequently writes on parenting issues. To purchase this article, contact one of these New York Times Syndicate sales representatives*". Then we can get the right collapsed typed dependencies:

nn(Downey-2, Charles-1)
nsubj(City-7, Downey-2)
cop(City-7, is-3)

...
nsubj(writes-14, writer-11)
advmod(writes-14, frequently-13)
rcmod(City-7, writes-14)
rcmod(writer-11, writes-14)
nn(issues-17, parenting-16)
prep_on(writes-14, issues-17
aux(purchase-2, To-1)
det(article-4, this-3)
dobj(purchase-2, article-4)
...

5.2 Sentence Compression

We obtain typed dependencies information by parsing every sentence with Stanford Parser which has done well on the sentence parsing and gained much attention.

For example, a candidate sentence is "*The government has issued a series of regulations and measures to improve the country's coal mine safety situation, the Xinhua said*."

After using the Stanford Parser, we got the dependency-based representations as followed:

det(government-2, The-1)
nsubj(issued-4, government-2)
aux(issued-4, has-3)
ccomp(said-23, issued-4)
det(series-6, a-5)
dobj(issued-4, series-6)
prep_of(series-6, regulations-8)
prep_of(series-6, measures-10)
conj_and(regulations-8, measures-10)
aux(improve-12, to-11)
xcomp(issued-4, improve-12)
det(country-14, the-13)
poss(situation-19, country-14)
amod(situation-19, coal-16)
nn(situation-19, mine-17)
nn(situation-19, safety-18)
dobj(improve-12, situation-19)
det(Xinhua-22, the-21)
nsubj(said-23, Xinhua-22)

Firstly, we construct a model for each of the dependencies:

$$modifier\left(keyword_1 - serial\ numbers_1,\ keyword_2 - serial\ numbers_2\right)$$

Simple form is:

$$modifier\left(w_1 - n_1,\ w_2 - n_2\right)$$

Secondly, we segment each sentence in the summary into clauses:

$$S = \{s_1, s_2, \ \ldots\ , s_n\}$$

$\boldsymbol{s_i}$: the ith clause of $\boldsymbol{S}$.

We have done a lot of research on the example materials and find that modifiers such as "nsubj/nsubjpass" and "dobj" are important which are shown in the front of the dependencies sequence. Additionally the clauses $\boldsymbol{s_i}$ (i=1,2, …, n) which contain w_1 and w_2 often can well express the main meaning of the sentence S. We develop our own strategy which extracts w_1 and w_2 from the line begining with "nsubj/nsubjpass/dobj". Here is the detailed information.

After we extracted the keywords w_1, w_2 and the serial numbers n_1, n_2, we combine the clauses $s_1$and s_2 which contain both $w_1$and w_2. There are several specific situations.

1) if w_1, w_2 are in the same clause, then we need to extract s_1 only;

2) if w_1, w_2 are not in the same clause and $n_1<n_2$, then we need to extract both s_1, s_2 , and concatenate s_2 after s_1 ;

3) if w_1, w_2 are not in the same clause and $n_2<n_1$, then we need to extract both s_1, s_2 , and concatenate s1 after s_2 .

For the above example, we should extract the words "issued-government, issued-series" and serial numbers "4～2, 4～6". Since the four words are in the same clause, we just extract the first clause, "The government has issued a series of regulations and measures to improve the country's coal mine safety situation".

We didn't do further operations for clause compression, because the results are not satisfied. Table 1 shows two examples.

Overall, the result of sentence compression algorithm achieved our expectation, it can recognize and remove some relative dates and "said" clauses such as "on Tuesday" or "the President said", which often don't appear in a summary.

But due to the diversity of language expression, Stanford Parser may give an incorrect representation of typed dependencies for a sentence. Our program is also not perfect enough; sometimes we may get an undesirable result. There is still a long way for us to go through.

5.3 Final processing

After the above steps, some of the summaries have reached the overall length limit, but others are not. We need to do more modifications to achieve the goal of 100 words, including removing some less important sentences or unimportant modifiers.

Although all the sentences in the original summary contain rich information, in order to meet the 100 words constraint, we have to remove some sentences. We think that shorter sentences have less information than the longer ones. They should be

deleted if necessary. Here is the algorithm:

1) define an instance object of map class: Map <String, Integer> weizhi_length, where "String" represents the sentence itself while "Integer" stands for its length.

2) put all these sentences into it according to the length's ascending arrangement.

3) while $(s-(c-100)<=threshold_1)$, remove the sentence from the map, then $c=c-s$;
c: the length of current summary;
s: the length of the sentence being operated;
$threshold_1$: we assign 6 to it, this value is obtained by a series of observations.

4) reorder the sentences in the map, if ($c>threshold_2$) then remove the unimportant modifiers. Here we assign 105 to $threshold_2$; this value is also obtained by a series of observations.

5) In Table 2, we list some abbreviation examples to replace the long phrases. Repeat these steps until the summary satisfies the 100 words limit.

6) There are a few sentences starting with personal pronouns (He/She/You/Him/Her/They/Them, etc.) and demonstrative pronouns (This/These/That/Those, etc.), this may do harm to the readability of the summary. We move them to the other part of the summary.

Since we extract sentences from multi-documents without considering their semantic sequence, the readability of the summary may be a little poor. We will go on with it in future.

6. Update Summarization

We also participated in the update summarization track of TAC 2010, which is to write a 100 words limit summary of a set of newswire articles, under the assumption that the user has already read the earlier articles. The summaries will also be evaluated for readability, content (based on Columbia University's Pyramid Method)[13] and overall responsiveness. The update summary should have different content from the initial one, and more focuses on follow-up report.

We use the same method as the initial summarization track. The only difference is that we choose novel sentences that have not been contained in the initial summaries.

We assume all sentences in initial summaries as the candidate sentences of update summary, and we will choose the new sentences that have least similarity with these candidate sentences. We use the formula (4-3) to calculate the similarity. If the similarity is bigger than the threshold, we will choose another one until the update summary has got ten sentences. This method can avoid duplicate information effectively.

7. Results and Discussion

In the guided summarization track of TAC 2010 we submit two runs. The first one is mainly considering the feature of keyword coverage, and the second one is mainly considering the feature of aspect coverage. The evaluation results show that our two runs are similar. We compare our two summarization results with the best one in 43 runs, which contain 41 runs from 23 participants and two baseline runs. The following figures 2,3 show the result.

Our result is not very optimistic. The following factors may infect the outcome. The first one is our clustering algorithm which plays an important role in determining the summary. It tends to cluster the same sentences into a group, but these sentences may not contain the most important content. The second one is the knowledge base, which is limited in coverage, authority and size. The third one is our sentence compression method which may prune some critical information. The last one is we didn't use the title and time information which appear in the original document set.

8. Conclusions and Future Research

This report introduces the details of a multi-document summarization system for both initial and update summaries. We construct a thesaurus to guide the sentence extraction, use hierarchical clustering to group sentences with similar content and propose sentence compression with Stanford Parser to condense the summary. The experiment shows the effectiveness of the algorithm. Although our multi-document summarization has met the basic requirements of the TAC evaluation, it needs to be improved. To generate better summaries, we can improve our clustering algorithm and utilize the title information and various named entities in the original document set. We can also improve our sentence compression method.

Table 1 Example of compressed sentences

Original sentence	*Compressed sentence*
The room in which the identification was taking place was small, and it was taking time for the relatives to file through, she said.	*The room in which the identification was taking place was small, and it was taking time for the relatives to file through.*
Shortly after takeoff Sunday morning, the Helios pilot radioed to ground control in Lanarca that he was having trouble with the air conditioning, Greek officials said.	*The Helios pilot radioed to ground control in Lanarca that he was having trouble with the air conditioning.*

Table 2 Example of abbreviations

Original phrases	*Abbreviations*
Royal Australian Air Force	*RAAF*
International Civil Aviation Authority	*ICAA*
Traditional Chinese Medicine Hospital	*TCMH*
China National Petroleum Corporation	*CNPC*

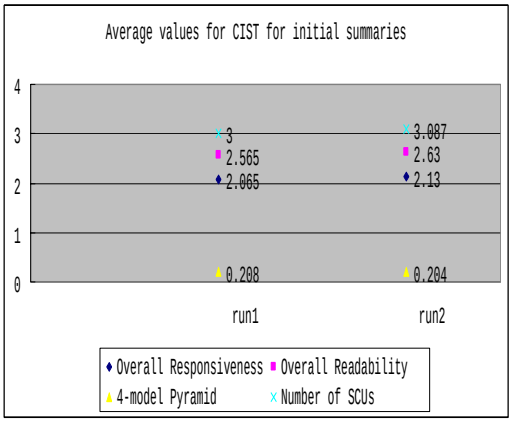


Figure 2 average values for initial summaries

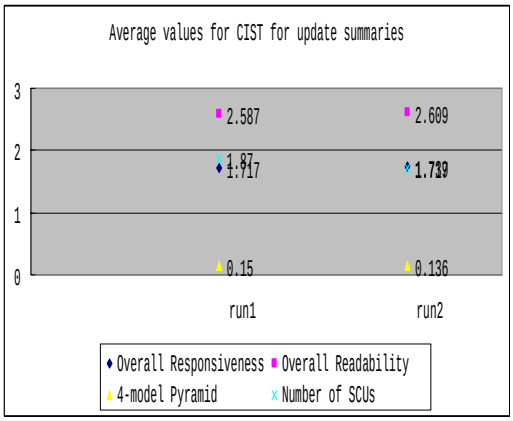


Figure 3 average values for update summaries

References

[1] Gong, Y., & Liu, X. (2001). Generic text summarization using relevance measure and latent semantic analysis. In Proceedings of the 24th annual international ACM SIGIR conference on research and development in information retrival (SIGIR'01) (pp. 19–25). New Orleans, USA.

[2] K. Knight and D. Marcu. Summarization beyond sentence extraction: a probablistic approach to sentence compression. Articial Intelligence, pages 91{107, 2002.

[3] H. Jing and K. McKeown. Cut and paste based text summarization. In Prodeedings of NAACL 2000.

[4] D. Radev, H. Jing, M. Stys, and D. Tam. Centroid-based summarization of multiple documents. Information Processing and Management, pages 919-938, 2004.

[5] C.-Y. Lin and E. Hovy. From single to multi-document summarization: A prototype system and its evaluation. In Proceedings of ACL 2002.

[6] S. Park, J.-H. Lee, D.-H. Kim, and C.-M. Ahn. Multi-document summarization based on cluster using non-negtive matrix factorization. In Proceedings of SOFSEM 2007.

[7] D. Shen, J.-T. Sun, H. Li, Q. Yang, and Z. Chen. Document summarization using conditional random elds. In Proceedings of IJCAI 2007.

[8] J. Conroy and D. O'Leary. Text summarization via hidden markov models. In Proceedings of SIGIR 2001.

[9] R. Mihalcea and P. Tarau. A language independent algorithm for single and multiple document summarization. In Proceedings of IJCNLP 2005.

[10] S. Harabagiu and F. Lacatusu. Topic themes for multi-document summarization. In Prodeedings of SIGIR 2005.

[11] Knight, Kevin and Daniel Marcu. 2002. Summarization beyond sentence extraction: a probabilistic approach to sentence compression. Artificial Intelligence 139(1):91–107.

[12] Lin, Chin-Yew. 2003. Improving summarization performance by sentence compression — a pilot study. In Proceedings of the 6th International Workshop on Information Retrieval with Asian Languages. Sapporo, Japan, pages 1–8.

[13] R.J. Passonneau, A. Nenkova, K. McKeown, and S. Sigelman, Applying the Pyramid Method in DUC 2005.

[14] Dan Zhou, Lei Li, "Research In Sub-topic Based Multi-document Summarization", 2008

The CIST Evaluation System at TAC 2010 AESOP

Qing Zhao, Hongyan Liu, Ying Xiong, Lei Li, Caixia Yuan
Center for Intelligence Science and Technology
Beijing University of Posts and Telecommunications
Beijing, 100876, China

Abstract

In this report, we compute two sets of numeric summary-level scores separately with different algorithms. In the "All Peers" case, we get the numeric score for each peer summary, including the model summaries. Experiments show that it can well distinguish model summaries from automatic ones. In the "No Models" case, we get the similarity score between each automatic summary and model summary by looking up the Roget's Thesaurus. Finally, we analyze the evaluation results of our two evaluations and suggest some possible improvements.
Keywords: All Peers, No Models, sentence coverage, document coverage, aspect coverage, similarity

1 Introduction

This is the first time we attend Text Analysis Conference. The actual AESOP task is to produce two sets of numeric summary-level scores:

•All Peers case: a numeric score for each peer summary, including the model summaries. It is intended to focus on whether an automatic metric can differentiate between human vs. automatic summarizers.

•No Models case: a numeric score for each peer summary, excluding the model summaries. It is intended to focus on how well an automatic metric can evaluate automatic summaries.

We built a system for TAC 2010 AESOP Task with two sets of numeric summary-level scores separately calculated by different algorithms.

We get all the test data from the TAC 2010 Summarization Track web page and submit four runs to NIST.

In "All Peers" case, our evaluation system uses the set of source documents as reference instead of the four model summaries TAC provided. The main method we used to evaluate automatic summaries and model summaries are almost the same, but since model summaries' expression may be diversified and flexible, when we compute the sentence and document coverage scores, we not only use words but also considered their synonyms, which are expanded with Roget's thesaurus.

In "No Models" case, we use model summaries as the reference to evaluate automatic summaries. For each document set (including 43 files), four human summaries were created as the ideal summaries. All we should do is to compute the similarity between each automatic and model summary.

2 The "All Peers" case

The "All Peers" evaluation aims to evaluate the content, readability and overall responsiveness of all summaries, including the automatic and model summaries. In this case, our evaluation system uses the source documents as reference instead of the four model summaries that TAC provided. The evaluation tends to distinguish automatic summaries from model ones.

2.1 Framework

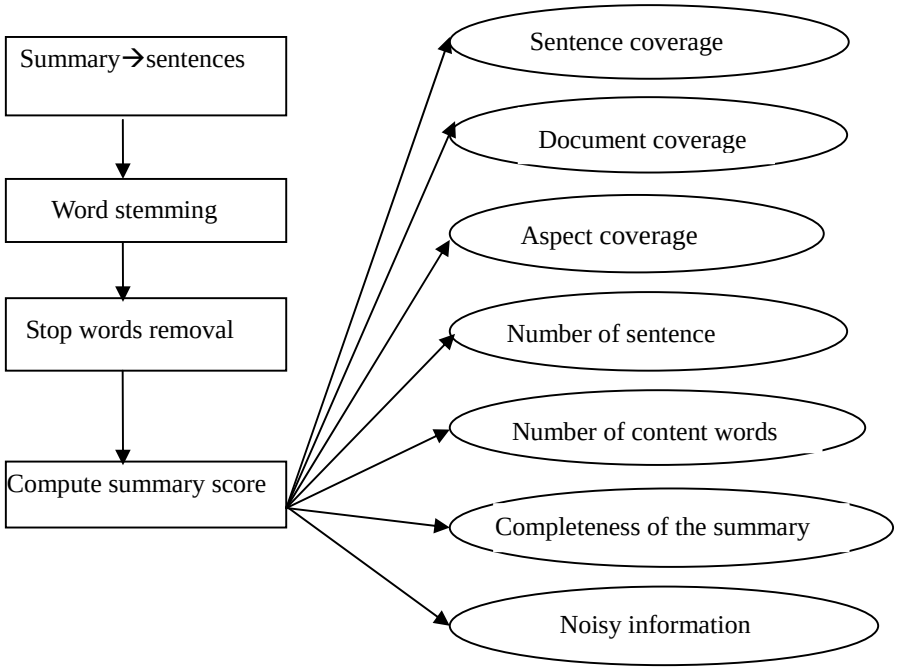


Figure 1 general evaluation framework for "All Peers" evaluation

Figure 1 shows the framework of the "All Peers" evaluation. We use the source documents as reference, but one problem is that these documents that TAC provided have many tag information which we don't need in the evaluation. We do preprocessing at first, including tag deletion, document segmentation, word stemming, and stop words removal.

To evaluate each summary, we want to find out whether or not the words in it are contained in the relevant ten original documents. In order to search quickly, we construct two inverted index files based on information retrieval, one is word-sentence index file and the other is word-document index file.

We evaluate all summaries according to the following features: sentence and document coverage of the source documents, number of the sentences and number of the content words in summary, completeness of the summary within 100 words, noisy information in the summary and aspect coverage for each topic.

2.2 Automatic Summaries' Evaluation

For all the content words in the summaries, we look up the word-sentence and word-document inverted index files respectively, then compute the coverage of the words in one sentence.

We assume that one summary is comprised of sentences $\{s_1, s_2, \ldots s_m\}$ and one sentence s_i is comprised of words$\{w_1, w_2, \ldots w_n\}$. For w_j in the sequence$\{w_1, w_2, \ldots w_n\}$, we count the sentences set sw_j which the word w_j has covered and we will get a set of $\{sw_1, sw_2, \ldots sw_n\}$ in the end, $ns_i=sw_1 \cup sw_2 \ldots \cup sw_n$, and $num_i=|ns_i|$, num_i is the total number of sentences that all words in one sentence has covered. max_s is the max number of the set $\{num_1, num_2, \ldots num_m\}$.

We calculate the sentence coverage score by formula (2-1).

$$s_{scovi} = num_i / \max_s \qquad (2\text{-}1)$$

Similarly, we also compute the score of document coverage. For w_j in $\{w_1, w_2, \ldots w_n\}$, we count the documents set d_j of the word w_j has covered and we will get $\{d_1, d_2, \ldots d_n\}$, $d_i=d_1 \cup d_2 \ldots \cup d_n$,

and $num_{di}=|d_i|$. The score of documents coverage is calculated by formula (2-2).

$$s_{d\,cov\,i} = num_{di} / 10 \qquad (2\text{-}2)$$

s_{dcovi} is the score of documents coverage, num_{di} is the total number of the d that all words in one sentence have covered, and 10 is for that one source document set has ten documents.

The third feature we considered is aspect coverage. We calculate the coverage of aspects according to the knowledge base, which is constructed by extracting keywords from TAC sample document sets for all the five topics which contain tagged information units for the required aspects. We obtain an original version of the knowledge base for each topic with many required aspects and each aspect corresponds to a keyword list. Then we expand these keywords with thesaurus of Britannica Online Encyclopedia (http://www.britannica.com/bps/thesaurus?query=good) using a simple meta search engine. To ensure the quality of synonyms, in the final version of the knowledge base, we improve it through further manual screening. If the word wj in the knowledge base, the aspect value aj is 1, otherwise aj is 0. The formula is (2-3).

$$s_{aspi} = \frac{\sum_{j=1}^{n} a_j}{num_{asp}} \qquad (2\text{-}3)$$

s_{asp} is the score of one sentence covering aspects of the category, num_{asp} is the number of the aspects that a category should contain.

Then we calculate the sentence score by the formula (2-4).

$$sen_i = s_{s\,cov\,i} * s_{d\,cov\,i} * s_{aspi} \qquad (2\text{-}4)$$

In “All Peers” case, we also evaluate the readability and overall responsiveness of the summary. We consider the number of the sentences and the content words in one summary. We think that in one 100 words summary there should be appropriate number of sentences, and the length of the sentence influence the comprehension of the readers directly. It would be better containing 4-8 sentences, and too many or too few sentences all will influence the overall responsiveness of the summaries. The score is defined as s_s.

Table 1 the score of s_s

number of sentences: m	value of s_s
m==1	0
m>1&&m<4	0.9
m>=4&&m<=8	0.95
m>8	0.9

At the same time we expect more content words which can express the topic information in a good summary. We design several score levels according to the number of words. We have done an experiment on model summaries and found out that most model summaries have 45-80 content words, and then we evaluate all summaries considering this feature. The number of the words in one summary fall into this range will be given a relatively higher score denoted as s_w.

Table 2 the score of the s_w

number of the content words: n	value of s_w
n<45	0.85
n>45&&n<80	0.9
n>80	0.85

A survey of all summaries show that some of them have noisy information such as “<p>”、“<BODY>”、“<DOC>”、“<TRAILER>”and so on, they have nothing to do with the topic contents. We compute the number of tags a summary contains for the score s_n.

Table 3 the score of s_n

number of tags: tags	value of s_n
Tags = 0	1
tags>0 && tags<=5	0.85

Tags>5 && tags<=10	0.8
Tags>10 && tags<=15	0.75
Tags>15 && tags<=20	0.7
Tags>20	0.65

The summary completeness is also a feature we have considered. The most important characteristic is some summaries contain incomplete sentences, which will influence the overall quality. And the score of the completeness of one summary is defined as s_c.

Table 4 the score of s_c

Completeness of the summary	score
true	1
false	0.8

Finally, we calculate the score of one summary s_{sum} by formula (2-5).

$$s_{sum} = \frac{1}{m}\sum_{i=1}^{i=m} sen_i * s_s * s_w * s_n * s_c \qquad (2\text{-}5)$$

2.3 Model Summary's Evaluation

Model summaries have the diversity and flexibility in language expression. When we compute the sentence and document coverage scores, we not only use words in model summary, but also considered their synonyms according to Roget's thesaurus. Experiments show that using the synonyms is effective in model summary evaluation.

Here is the flowchart about Model summary's evaluation.

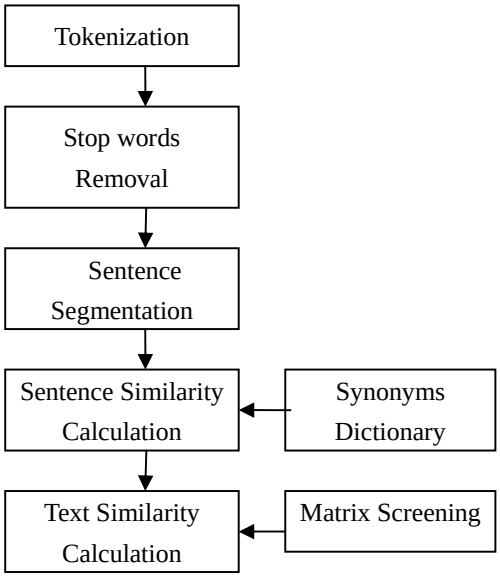


Figure 2 No Models Evaluation based on Synonyms Dictionary lookup

The Roget's thesaurus is consisted of two files: the "1068-index.txt" and the "1068-body.tx". The relationship between the tow files is demostrted in table 5.

Table 5 the relationship between index file and body file

10681-index.txt	10681-body.txt
$word_1$ ($phrase_1$) $synonyms_{11}$ $index_{11}$ $synonyms_{12}$ $index_{12}$ … $word_i$ ($phrase_i$) $synonyms_{i1}$ $index_{i1}$ $synonyms_{i2}$ $index_{i2}$ … …	1. $word_j$ 2. . . . $index_{11.}$ a list of words that have the same meaning with $word_1$ ($phrase_1$) . . . $index_{i1.}$ a list of words that have the same meaning with $word_i$ ($phrase_i$) . . .

3 The "No Models" case

We've got 4324 documents zipped in one folder from TAC website as the evaluation materials, including model summaries and automatic ones. Since the "No Models" case is intended to focus on how well an automatic metric can evaluate automatic summaries, we first separate the automatic summaries from model ones.

Each document set includes 43 automatic summary files and four human summaries which were created as the "ideal" summaries. All we should do is to compute the similarity between each automatic and model summary.

3.1 Framework

We develop a series of steps to evaluate automatic summaries. Firstly, some preprocessing should be done, including tokenization and stop words removal. Then we mainly pay attention to sentence similarity and text similarity calculation. Figure 2 which is showed above is a flowchart of the evaluation.

3.2 Proposed Methods

The main idea of automatic evaluation of summaries based on synonyms dictionary is: looking up the dictionary to find out whether the words in automatic summary and the model summary are synonymous. Synonyms have higher similarity value. We calculate the similarity between sentences based on words; then we get the text similarity by screening the sentence similarity matrix.

3.2.1 Sentence similarity calculation

Here is the formula (3-1) for the sentence similarity proposed in paper[1].

$$sim(s_i, s_j) = 2 * samewc(s_i, s_j) / [len(s_i) + len(s_j)]$$

(3-1)

$samewc(s_i, s_j)$: the number of words with the same meaning in sentence s_i and s_j, which is counted by looking up the synonyms dictionary, more details will be give in section 3.2.2;

$len(s_i)$: the length of sentence s_i;

s_i: the i_{th} sentence in the automatic summary D_1;

s_j: the j_{th} sentence in the model summary D_2.

Following the same idea, we proposed our own formula. We think that comparing to synonyms, the same words in automatic summary and the model summary contribute more to the similarity. Our metric is calculated in different conditions as shown in formula (3-2):

if $w_1 = w_2$, $samew = 1$;

else if w_1 and w_2 are synonyms, $samew = 0.9$;

$length(s_i)$: the length of sentence s_i without stop words;

$$sim(s_i, s_j) = 2 * samew(s_i, s_j) / [length(s_i) + length(s_j)]$$

(3-2)

Assume that there are m sentences in D_1, n sentences in D_2, then we will get a $m*n$ matrix $M(D_1, D_2)$:

$$M(D_1, D_2) = \begin{pmatrix} sim(s_{11}, s_{21}) & \cdots & sim(s_{11}, s_{2n}) \\ \vdots & \ddots & \vdots \\ sim(s_{1m}, s_{21}) & \cdots & sim(s_{1m}, s_{2n}) \end{pmatrix}$$

(3-3)

S_{i1}: the i_{th} sentence in document D_1;

S_{i2}: the i_{th} sentence in document D_2.

3.2.2 Synonyms Dictionary lookup

Roget's Thesaurus is our synonyms' dictionary. There are two files, "10681-index.txt" and "10681-body.txt". Their relationship is shown in Table5.5.

Since we only need single word synonyms, we'd better remove phrases from the index file so as to get a better time and space efficiency.

The algorithm for judging whether word x from automatic summary and word y from model summary are synonyms is as followed:

(1) set array_$index[\ ]=\{w_1\ n_{11}\ldots n_{1k}, \ldots, w_i\ n_{i1},\ldots n_{im}, \ldots\}$;

w_i: the i_{th} word in the index file;

n_{im}: the index of word w_i's m_{th} meaning;

set $array\_body[\ ]=\{list_1, \dots, list_j, \dots\}$;

$list_j$: a list of words in "10681_body" which have the same meaning with word w_j in "10681_index" ;

(2) if $x=w_i$, then extract indexes $n_{i1},\dots n_{im}$, then go to (3), else return false;

(3) for each index, take n_{i1} for example, if y is in array $body[n_{i1}]$, return true, then break; else continue searching

3.2.3 Text similarity calculation

In section 3.2.1, we get the similarity matrix $M(D_1, D_2)$, here we should sift the elements in M to find a queue of sentence similarities, and then get the average value as the similarity between the automatic summary and the model summary.

Detailed steps of the method are as followed:

(1) traverse the matrix, find the maximum value $simS_{max}$ and put it into queue S; then set all the elements which are in the same row and column with $simS_{max}$ to 0, at last we get a new matrix $M(D_1, D_2)$, go to step (2);

(2) if the matrix is empty or the elements are all 0, go to step (3), else go to step (1);

(3) we get the final queue $S=\{simS_{max1}, simS_{max2}, \dots, simS_{maxk}\}$.

(4) the similarity between D_1 and D_2 is calculated by formula (3-4):

$$sim(D_1, D_2) = \frac{1}{k}\sum_{i=1}^{k} simS_{\max_i} \qquad (3\text{-}4)$$

4 Submissions and Results

We submitted four runs to NIST, and each run evaluates "All Peers" and "No Models" separately.

Run 1: In "All Peers" case, we didn't consider the aspect coverage. We considered all the content words in each peer summary while expanding synonyms of the content words in model summaries. In "No Models" case, we look up the synonyms dictionary without considering the aspect coverage.

Run 2: In "All Peers" case, we didn't consider the aspect coverage. We considered the number of all nouns and verbs without expanding synonyms in model summaries. In "No Models" case, we didn't look up the synonyms dictionary but considered the aspect coverage.

Run 3: In "All Peers" case, we add the aspect coverage and considered all the content words in each peer summary while expanding synonyms for model summaries. In "No Models" case, we look up the synonyms dictionary and considered the feature of aspect coverage.

Run 4: In "All Peers" case, we considered the aspect coverage feature and the number of all nouns and verbs, but we didn't expand synonyms for words in model summaries. In "No Models" case, we didn't look up the synonyms dictionary while considering the feature of aspect coverage.

Finally our submissions are tested on the TAC 2010 dataset. The AESOP evaluation results of summary A and B are shown in Figure 3 and 4.

In both figures, "Best" means the best result among all 27 submissions. "Our best" means our best result among the 4 submitted runs. The left part of each figure shows the correlations with Pyramid for each run, and the other is correlations with Responsiveness.

5 Conclusions and Future Work

We can see that our "All Peers" method performs better than "No Models" method in AESOP evaluation. It gets a higher score on most "All Peers" correlation items. For example, in Figure 3, every correlation's score in "All Peers" case is higher than that in "No Models" one.

In both figures, the overall trend of data in "All Peers" case is closer to the "Best" one than that in "No Models" case.

Scores are higher in Figure 3 in both "All Peers" and "No Models" case than that in Figure 4,

which means summary A is better than summary B.

Although our systems' overall levels are not ideal, our Spearman score is much better than other teams.

There is a long way for us to improve the evaluation systems, especially in the following aspects:

1). improve the similarity calculation algorithm;

2). consider the linguistic quality of the summaries to be evaluated;

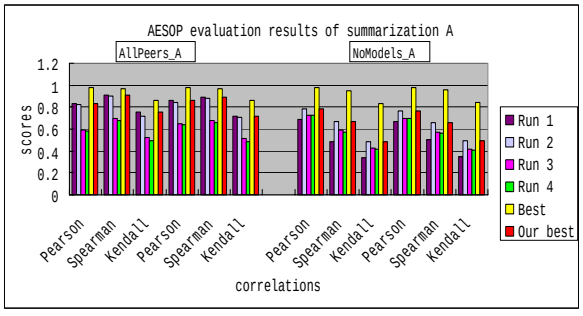
3). make the synonyms searching algorithm more effective.

Hope that we could get a progress in summary evaluation next year.

results	Correlations with Pyramid			Correlations with Responsiveness		
	Pearson	Spearman	Kendall	Pearson	Spearman	Kendall
Run 1	0.835	0.914	0.725	0.857	0.89	0.717
Run 2	0.82	0.897	0.719	0.841	0.883	0.706
Run 3	0.595	0.698	0.52	0.652	0.675	0.512
Run 4	0.583	0.675	0.496	0.641	0.659	0.488
Best	0.975	0.965	0.862	0.977	0.97	0.86
Our best	0.835	0.914	0.752	0.857	0.89	0.717

results	Correlations with Pyramid			Correlations with Responsiveness		
	Pearson	Spearman	Kendall	Pearson	Spearman	Kendall
Run 1	0.688	0.483	0.34	0.666	0.504	0.347
Run 2	0.786	0.672	0.487	0.763	0.661	0.491
Run 3	0.725	0.59	0.422	0.696	0.575	0.419
Run 4	0.725	0.57	0.416	0.697	0.557	0.41
Best	0.978	0.948	0.836	0.979	0.958	0.841
Our best	0.786	0.672	0.487	0.763	0.661	0.497

(a)



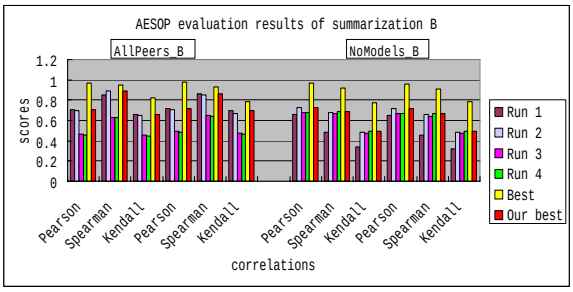
(b)

Figure 3 "All Peers" and "No Models" results of summary A

results	Correlations with Pyramid			Correlations with Responsiveness		
	Pearson	Spearman	Kendall	Pearson	Spearman	Kendall
Run 1	0.704	0.853	0.659	0.716	0.864	0.666
Run 2	0.683	0.884	0.653	0.704	0.854	0.67
Run 3	0.484	0.631	0.459	0.495	0.645	0.478
Run 4	0.452	0.632	0.442	0.483	0.638	0.461
Best	0.969	0.946	0.821	0.978	0.928	0.787
Our best	0.704	0.884	0.659	0.716	0.864	0.666

results	Correlations with Pyramid			Correlations with Responsiveness		
	Pearson	Spearman	Kendall	Pearson	Spearman	Kendall
Run 1	0.655	0.481	0.343	0.648	0.455	0.321
Run 2	0.723	0.674	0.485	0.715	0.662	0.486
Run 3	0.674	0.663	0.474	0.663	0.637	0.47
Run 4	0.676	0.686	0.49	0.665	0.664	0.481
Best	0.964	0.921	0.771	0.96	0.91	
Our best	0.723	0.686	0.49	0.715	0.664	0.481

(a)



(b)

Figure 4 "All Peers" and "No Models" results of summary B

References

[1] HUANG Li-qiong, HE Zhong-shi, ZHANG Jie-hui, Automatic summarization evaluation method based on similarity of text, Application Research of Computers， 2007,24(8)

[2] Rebecca J. Passonneau, Ani Nenkova, Kathleen McKeown, Sergey Sigelman, Applying the Pyramid Method in DUC 2005
Columbia University Computer Science Department New York

[3] Liu Yin, Li Bicheng The Overview and Prospect of Automatic Summarization Evaluation Department of Information Science, Information Engineering Institute, Information Engeering University, Zhengzhou 450002

[4] Li-Qing Qiu, Bin Pang Analysis of automated

evaluation for multi-document summarization using content-based similarity State Key Lab. of Software Development Environment, Beihang University, 100083

[5] Eduard H, Lin C Y, Zhou L, Junichi F. Automated Summarization Evaluation with Basic Elements. In Proceeding of the Fifth Conference on LREC, Genoa, Italy.2006

[6] Donaway R, Drummey K, Mather L.A Comparison of Rankings Produced by Summarization Evaluation Measures. In Proceeding of ANLP/NAACL Workshop on Automatic Summarization, pages 69-78, 2000

[7] Saggion H, Radev D, Teufel S, et al. Meta-evaluation of Summaries in a Cross-lingual Environment Using Content-based Metrics In Proceeding of the 19th ICCL ,pages 849855,2002

[8] Lin C Y.ROUGE: A Package for Automatic Evaluation of Summaries In Proceeding of DUC, 2004.

[9] Lin C Y, Hovy E. Automated Evaluation of Summaries Using N-gram Co-Occurrence Statistics In Proceeding of HLTC, pages 71-78, 2003