TAC 2008 Update Summarization Task of ICL

Sujian Li, Wei Wang, Chen Wang
Inst. of Computational Linguistics, Peking University
{lisujian, wwei, goldeneagle}@pku.edu.cn

Abstract

The update task of multi-document summarization aims at automatically generating the summaries of some event developing with time going. Based on our previous system of multi-document summarization, we summarize the document sets with or without history. In our previous system, the design of features is the important part. Here, in order to adapt to the updating task we introduce a new ‘filtering’ feature. With history, filtering features are calculated to exclude those sentences which are similar to the “history”. The principle of designing filtering features is to distinguish the current documents from the previous documents, and reflect the main idea of current documents. In this work, we combine two kinds of similarity metrics and two computing strategies to get the filtering feature value. The experimental results will show which combination is more effective to catch the developing process of an event.

1. Introduction

The TAC[1] 2008 update summarization task is similar to that in DUC 2007 [Hoa 2007], which aims at generating short (not more than 100 words) fluent multi-document summaries of news articles under the assumption that the user has already read some earlier articles. The earlier articles are called the “history” in this paper. The only difference is that TAC 2008 divides the articles of one topic into two sets while DUC 2007 divides into three sets. For each topic, the first document set is summarized just like the main summarization task in DUC 2007, except that the document number and the summary length limit are different. The second document set needs to be summarized with consideration of its history, that is, the first document set. Our previous system for DUC main task is designed with feature-based sentence-extractive framework. In this paper, we adopt the same system framework and introduce a new ‘filter’ feature to adapt to the update task.

In our feature-based system, various features in the sentence are used to judge whether the sentence should be appropriately included in the summary. Feature weights can be tuned with machine learning methods [Li 2007] or manually [Li 2006], which depends on whether the training data is available. Then for the update summarization task, the key is to design a new feature to reflect that the summary is summarized with the history known. That is, the function of the feature is to avoid selecting those sentences which are similar to the history. Thus, we call the feature as ‘filtering’ feature, which is the focus of this paper.

The rest of the paper is organized as follows. Section 2 briefly describes our system design and some features used in previous system. Section 3 emphasizes on the design of the ‘filtering’ feature, which includes four kinds of designing methods. Section 4 presents the evaluation results of different designing methods. Section 5 shows the future work and concludes the paper.

[1]http://www.nist.gov/tac/

2. System overview

In this section, we first formalize the update task of summarization. For a given topic, all the articles are separated into several sets T_1, T_2, …, T_n. The articles in T_1, T_2, …, T_{i-1} are seen as the history of T_i ($i \geq 2$). When the articles in T_1 need to be summarized, there is no history. Then, the summarization task is just like the main summarization task of DUC. However, the articles in T_i ($i \geq 2$) are summarized with the assumption that the content in the history has been known. No matter whether the articles summarized have history, we adopt a uniform feature-based summarization framework. For the possible effect of history on the summarization result, we design a "filtering" feature for every sentence. When summarizing T_1, the values of filter feature are the same (=0) for all the sentences in T_1. When summarizing other set T_i ($i \geq 2$), the filtering feature is calculated to represent the different degrees of each sentence overlapping with the history.

Our summarization system is designed with the extractive framework. Important sentences are extracted and re-organized to form a summary. Thus, the whole system is mainly divided into three modules: text preprocessing, sentence extraction and post-processing. In text preprocessing, query and documents are segmented into sentences, and then we conduct POS tagging and named entity recognition for each sentence, preparing for the next feature extraction. The focus of sentence extraction module is on which feature to extract and how to rank the importance of each sentence with reference to their features. In the post-processing module, sentences with higher scores are extracted to compose of the summary with MMR method. In the following subsections, we will overview the ranking method and the features used in the system.

2.2 Ranking method

Here we rank the sentences with a linear combination of features. That is, each sentence is assigned a score which cumulates the impacts of each feature. The impact of each feature is represented by its weight, which are tuned by experience. The formula is as follows.

$$\mathrm{Score}_s = \sum w_i f_i$$

Where s means a sentence, f_i means the feature value while w_i indicates the weight of the feature f_i set experimentally.

2.2 Features Overview

The sentences are ranked and assigned an importance score according to various features. This subsection mainly overviews the features which are designed for a query-focused summarization system without considering history.

(1) Word Matching Feature

$$f_{word}(s) = \sum_{t_j \in s} \sum_{t_i \in q} same(t_i, t_j)$$

where f is the feature value, q is the topic description. The function $same(t_i, t_j) = 1$ if $t_i = t_j$, and 0 otherwise

(2) Name Entity Matching Feature

$$f_{entity}(s) = | entity(s) \cap entity(q) |$$

where $| entity(s) \cap entity(q) |$ is the number of the named entities in both s and q. Here four classes of named entities (person, organization, location, date) are involved.

(3) Semantics Matching Feature

$$f_{wordnet}(s) = \sum_{t_j \in s} \sum_{t_i \in q} similarity(t_i, t_j)$$

where the function $similarity(t_i, t_j)$ is the lesk similarity function introduced in [Satanjeev 2002], which is WordNet-based [Christiane 1998]and scales the semantic relation between two words.

(4) Document Centroid Feature

$$f_{centroid}(s) = \sum_{t_j \in s} tfidf(t_j)$$

where $tfidf(t_j)$ is the *tf-idf* score of t_j in the whole data set [Radev 2000].

(5) Named Entity Number Feature

$$f_{entityno}(s) = | entity(s) |$$

where $| entity(s) |$ is the number of named entities in *s*.

(6) Stop Word Penalty Feature

$$f_{stopword}(s) = | stopword(s) |$$

where $| stopword(s) |$ is the number of the stop words in *s*.

(7) Sentence Position Feature

$$f_{position}(s) = 1 - \frac{i-1}{n}$$

where *n* is the total number of the sentences and *s* is the *i*th sentence in the document.

3. Filtering Feature

This year, the focus of our system is how to choose a new feature to adapt to the update task. This feature is optional which depends on whether the articles summarized have a “history”. When the articles with history are summarized using a sentence extraction method, more overlapped with the history a sentence is, less chance the sentence is selected. That is, a sentence which is similar to the history content should be filtered. Thus, we design a filtering feature for measuring the similar degree.

When a sentence is compared with the history, firstly we need consider the similarity of the sentence with each sentence in the history. Then it is also considered with which strategy the similarity values are used as the filtering feature value.

We calculate the similarity between two sentences with two kinds of metrics introduced as follows.

(1) Complex similarity metric

The first takes into consideration the unigrams, bigrams and syntactic functions of the words.

The unigram factor is considered with the formula:

$$Uni = \frac{U_{AB}}{\sqrt{U_A{}^2 + U_B{}^2}}$$

Where U_A and U_B represent the length of two sentences *A* and *B* respectively. U_{AB} represents the number of common words occurring in both sentences.

The bigram value is calculated with the following formula:

$$Bi = \frac{BI_{AB}{}^2}{\sqrt{BI_A{}^2 + BI_B{}^2}}$$

Where BI_{AB} means the number of common bigrams occurring in both sentences. In order to strengthen the effect of common bigrams, the BI_{AB} is squared. BI_A and BI_B represents the number of bigrams respectively in sentence *A* and sentence *B*.

The syntactic functions of words are considered with:

$$Syn = \frac{VB_{AB}{}^2}{VB_A + VB_B} \times \frac{C_N \times NN_{AB}}{NN_A + NN_B} + \frac{NN_{AB}{}^2}{NB_A + NN_B} \times \frac{C_V \times VB_{AB}}{VB_A + VB_B} - \frac{NN^2}{NN_A + NN_B}$$

Where VB_{AB} means the number of common verbs occurring in both sentences, NN_{AB} means the number of common nouns in both sentences, VB_A and VB_B respectively represent the verbs in two sentences while NN_A and NN_B represent the nouns.

Finally, these three factors are combined linearly to get the similarity metric:

$$S_{A,B} = \alpha \cdot Uni + \beta \cdot Bi + \gamma \cdot Syn$$

Where α, β and γ are the weights of these three factors respectively. $S_{A,B}$ is the similarity value of two sentences *A* and *B*.

2) Simple similarity metric

The second metric adopts the simple cosine distance formula. Each sentence is represented by a vector of *tf*idf* of the words. Then the similarity is computed as:

$$S_{A,B} = \frac{V_A . V_B}{|V_A| * |V_B|}$$

Where V_A and V_B are two sentence vectors respectively.

Because a sentence is computed the similarity value with each sentence in the history documents, it is another problem how to get the final filtering feature with these similarity values. Here we use two kinds of strategies: Maximum and Average. The maximum strategy means that the filtering feature takes the maximal value of the similarity values. That is,

$$F_{filter} = \max_{s-history} S_{s-history,s}$$

Where *s-history* means a sentence in the history documents, and $S_{s\text{-}history,s}$ represents the similarity value between *s-history* and the current sentence *s*.

Using the average strategy, the filtering feature takes the average of the similarity values. The formula is:

$$F_{filter} = \underset{s-history}{avg}\ S_{s-history,s}$$

With combining the two similarity metrics with the two computation strategies, we can get four kinds of results as in Table 1. Three of them are our submitted runs which are named Run1, Run2, Run3. The remaining one is called Run4.

	Maximum strategy	Average strategy
Complex similarity	Run1	Run3
Simple similarity	Run2	Run4

Table 1: four kinds of results

4 Evaluations

TAC 2008 test datasets comprises approximately 48 topics. Each topic has a topic statement (title and narrative) and 20 relevant documents which have been divided into 2 sets: Document Set A and Document Set B.

NIST assessors created 4 reference summary for each set of articles. All submitted systems are either manually or automatically evaluated, including linguistic quality, responsiveness, ROUGE-2, ROUGE-SU4[Lin 2004], and Pyramid. Each system is required to submit no more than three runs.

We submitted three runs: Run1, Run2 and Run3. The reason we submitted these three runs is, that our intuition is the maximum strategy is better and we favor the maximum strategy. Both Run1 and Run2 adopt the maximum strategy. The maximum strategy represents that the importance of a sentence will be reduced only if it is similar to any sentence in the history, while the average strategy stresses that the sentence will be ignored when it is similar to the history on the whole.

Table 2 illustrates the automatic evaluation results of our system and the best submitted system. From this table, it is surprising to find that the average strategy gets a better result than the maximum strategy for the simple similarity metric. It is the same of the manual evaluation results, which are listed in Table 3.

	R-2	R-SU4
Top1 system	0.10382 (0.09530–0.11302)	0.13625 (0.12875–0.14402)
Run1	0.08312 (0.07480–0.09194)	0.11893 (0.11240–0.12609)
Run2	0.08047 (0.07307 –0.08797)	0.11732 (0.11019–0.12521)
Run3	0.08278 (0.07463 –0.09089)	0.11621 (0.10943–0.12291)

Table 2: Automatic Evaluation in TAC 2008

	Pyramid	Ling. quality	resp
Top 1 system	0.336	3.333	2.667
Run1	0.287	2.031	2.344
Run2	0.284	2.031	2.417

Table 3: Manual Evaluation in TAC 2008

Then we experiment the filtering feature which combines the complexity similarity metric with the average strategy. The rouge results are illustrated in Table 4, compared with Run1 which adopts the maximum strategy. It follows that that the results of Run4 are obviously better than Run1, Run2 and Run3. It proves that the filtering feature is designed more reasonably when considering more factors and using the average strategy.

	R-2	R-SU4
Run1	0.08312 (0.07480–0.09194)	0.11893 (0.11240–0.12609)
Run4	0.08957 (0.08023 -0.10001)	0.12699 (0.11857 - 0.13645)

Table 4: Maximum strategy vs Average strategy for the complex similarity metric

5 Conclusions and Future Work

In this paper, we introduce a filtering feature for the update summarization system. In order to get the filtering feature, we adopt two similarity metrics and two kinds of computation strategies. The experimental results show that a filtering feature should consider more factors not limited to the simple statistics of words. It is also concluded that it is better to take the average strategy to select sentences.

In our future work, we will focus on considering more factors to design the filtering feature. And some machine learning methods will be experimented on the update task.

Acknowledgements

This work is supported by NSFC programs (No: 60603093 and 60875042), and 973 National Basic Research Program of China (2004CB318102).

References

Christiane, F., 1998. WordNet: an Electronic Lexical Database. MIT Press.

Hoa T. D.. *Overview of DUC 2007*. Document Understanding Conference 2007 http://duc.nist.gov, 2007.

Li, S.J. Ouyang, Y., Sun, B, Guo, Z.L., 2006. Peking University at DUC 2006. In Proceedings of DUC2006.

Li, S.J., OuYang, Y., Wang, W., Sun B., 2007. Multi-document Summarization Using Support Vector Regression, In Proceedings of DUC 2007.

Lin.C.Y., 2004. ROUGE: A Package for Automatic Evaluation of Summaries. In Proceedings of the Workshop on Text Summarization, Barcelona. ACL.

Radev, D. R., Jing, H.Y., and Budzikowska, M., 2000. Centroid-Based Summarization of Multiple Documents: Sentence Extraction, Utility-Based Evaluation, and User Studies. Proceedings of the 1st Conference of the North

American Chapter of the Association for Computational Linguistics, Seattle, WA, April 2000.

Satanjeev B., Ted P., 2002. *An Adapted Lesk Algorithm for Word Sense Disambiguation Using WordNet*. In Proceedings of the Third International Conference on Computational Linguistics and Intelligent Text Processing (CICLING-02). pages 136-145.