Comparison of feature usage at TSC-3 summarization tasks

Chikashi NOBATA† Satoshi SEKINE‡
Kiyotaka UCHIMOTO† Hitoshi ISAHARA†

†Computational Linguistics Group
National Institute of Information and Communications Technology
3-5 Hikaridai Seika-Cho Soraku-Gun
Kyoto 619-0289 Japan
{nova,uchimoto,isahara}@nict.go.jp

‡Computer Science Department
New York University
715 Broadway, 7th floor, New York, N.Y. 10003 USA
sekine@cs.nyu.edu

Abstract

We participated in two summarization tasks at the TSC-3. We have introduced categorization of feature values for our summarization system, which is based on sentence extraction technique. The categorized values were used as features for generating a decision tree. We compared our summarization system using the categorization of feature values with the one using linear combination of features in evaluation results of the TSC-3 tasks.

Keywords: *Multi-document summarization, Sentence extraction, Categorization of feature values*

1 Introduction

We have participated in the TSC evaluation for the third time. Our summarization system for tasks at the TSC-3 is a modified version of the system we developed for TSC-2 [8], which is based on sentence extraction technique.

Sentence extraction is one of the prominent methods for summarization[10, 6], which uses several features to estimate the importance of each sentence. Finding a new feature is one way to improve performance of sentence extraction. One observation, however, indicates that categorization of feature values is more appropriate to deal with the distribution of key sentences [9]. The performance of sentence extraction can possibly be improved by introducing the categorization of feature values without adding a new feature.

We have introduced categorization of feature values into our summarization system. The categorized feature values are combined using a decision tree. We compared the summarization system based on categorization of feature values with the one based on linear combination of feature values.

2 System overview

In this section, we briefly explain two summarization systems. One system (SysL) is to use linear combination of features for extracting key sentences, which are revised forms of those used in our previous system at TSC-2. The other system (SysC) is to categorize feature values and combine the categories to extract key sentences. We used decision trees for combining categorized feature values.

2.1 System with linear combination of features (SysL)

This system uses parameters to combine results of feature values in order to calculate the total score of a sentence. The total score of a sentence (S_i) is calculated using feature values ($Score_j()$) and parameters (α_j) as follows:

$$\text{Total-Score}(S_i) = \sum_j \alpha_j Score_j(S_i)$$

Some features have more than one scoring function, one of which was selected in the training stage. We explain how each feature value is set and used in the system in the following sections.

2.1.1 Sentence position

The system uses 'Sentence position' to set the significance of sentences. There are three different functions to handle sentence position. The first method is to give a score of 1 to the first N sentences and 0 to the others, where N is a given threshold for the number of sentences. That is, the score of the ith sentence *Score*(S_i) is:

$$\begin{aligned} \text{P1. } \textit{Score}_{\text{pst}}(S_i)(1 \leq i \leq n) &= 1 \quad (\text{if} \quad i < N) \\ &= 0 \quad (\text{otherwise}) \end{aligned}$$

where n is the number of sentences in a given article. The second method is to give the reciprocal of the sentence position; the score of ith sentence *Score*(S_i) is

$$\text{P2. } \textit{Score}_{\text{pst}}(S_i) = \frac{1}{i}.$$

These two methods are based on the hypothesis that the sentences in the beginning of the article are more important than those in the other part.

The third method is a modified version of the second one; the method checks the sentence position from the end of the article as well as the beginning:

$$\text{P3. } \textit{Score}_{\text{pst}}(S_i) = \max(\frac{1}{i}, \frac{1}{n-i+1}).$$

The method is based on the hypothesis that the sentences in both the beginning and the end of the article are more important than those in the middle.

2.1.2 Sentence length

The second feature used to set the significance of sentences is 'Sentence length.' The length here means the number of characters in the sentence. The first function returns the relative length of each sentence (L_i) to the maximum length of the sentence (L_{max}). Because we would like to set it uniformly in the whole document sets, we fixed the value of (L_{max}) to 200 in advance.

$$\begin{aligned} \text{L1. } \textit{Score}_{\text{len}}(S_i) &= \frac{L_i}{L_{max}} \quad (\text{if} \quad L_i \leq L_{max}) \\ &= 1 \quad (\text{otherwise}). \end{aligned}$$

The second function sets the score to a negative value as a penalty when the sentence is shorter than a certain length (L_{min}):

$$\begin{aligned} \text{L2. } \textit{Score}_{\text{len}}(S_i) &= 0 \quad (\text{if} \quad L_i \geq L_{min}) \\ &= \frac{L_i - L_{min}}{L_{min}} \quad (\text{otherwise}). \end{aligned}$$

Since we set L_{min} to 20 in the following evaluation, a sentence with 20 or less characters received a penalty score.

2.1.3 Tf*idf

The third method is based on term frequency and document frequency. The hypothesis here is that the more words that are specific to an article are in a sentence, the more important the sentence is. The target words are nouns (excluding temporal or adverbial nouns). For each of these nouns in a sentence, the system calculates the tf*idf score. The total score is the significance of the sentence. Word segmentation is performed by Juman3.61 [4].

When a set of documents is given in advance, our system counts the term frequency (*tf*) and the document frequency (*df*) for each word w, then calculates the tf*idf score as follows:

$$\text{tf*idf}(w) = \textit{tf(w)} \log \frac{\textit{DN}}{\textit{df(w)}}$$

where *DN* is the number of given documents. We used articles in the Mainichi and Yomiuri newspapers from 1998 and 1999 to count the document frequency.

The sentence score with tf*idf values of words is calculated with normalization[5]. When a document D is given, our system calculates the Euclidean norm of tf*idf values for all words in D (D_{norm}):

$$D_{\text{norm}} = \sqrt{\sum_{w \in D} \text{tf*idf}(w)^2}.$$

Then, the score for the ith sentence (S_i) in D is calculated as follows:

$$\textit{Score}_{\text{tf*idf}}(S_i) = \frac{1}{D_{\text{norm}}} \sqrt{\sum_{w \in S_i} \text{tf*idf}(w)^2}.$$

2.1.4 Headline

The fourth feature to set the significance of sentences is the headline of an article. The basic idea is that the more words in the sentence overlap with the words in the headline, the more important the sentence is. This function estimates the relevance between a headline(H) and a sentence(S_i) using the tf*idf values of words(w) except stop words in the headline:

$$\text{H1. } \textit{Score}_{\text{hl}}(S_i) = \frac{\sum_{w \in H \cap S_i} \text{tf*idf}(w)}{\sum_{w \in H} \text{tf*idf}(w)}.$$

We also calculated the feature value using only Named Entities (NEs) instead of the nouns. NEs were annotated by a pattern-based NE extraction program which has been developing to annotated extended NE categories presented at [12]. For NEs, only the term frequency was used because we judged that the document frequency for entities(e) was usually quite small

thereby making the difference between entities negligible:

$$\text{H2. } \textit{Score}_{\text{hl}}(S_i) = \frac{\sum_{e \in H \cap S_i} \textit{tf(e)}}{\sum_{e \in H} \textit{tf(e)}}.$$

2.1.5 Topic

The fifth feature is additional information about document sets. Each document set has topic description that explains the subject matter of the documents. The topic description was used in the similar way to the headline, the function with tf*idf values of words was used for topics(T). The equation of the method is as follows:

$$\textit{Score}_t(S_i) = \frac{\sum_{w \in T \cap S_i} \text{tf*idf}(w)}{\sum_{w \in T} \text{tf*idf}(w)}.$$

For training, we used 'Description' part in TSC-2 data as a topic of each document set.

2.1.6 Question

At TSC-3 tasks, there were about five questions for a short summary and about ten questions for long summary at each document set. To include sentences related with these questions (Q) into a generated summary, the function with tf*idf values of words was used as follows:

$$\textit{Score}_Q(S_i) = \max_{q_i \in Q} \frac{\sum_{w \in q_i \cap S_i} \text{tf*idf}(w)}{\sum_{w \in q_i} \text{tf*idf}(w)}.$$

The sentence score is the maximum value among scores with given questions so that a sentence related with at least one of the questions is picked up.

2.2 Parameters

This system calculates a significance score for all of the sentences, and sets the ranking of each sentence in descending order of score. The order of the extracted sentences is the same as in the original articles when the system outputs a summary.

The training data we used at the TSC-3 tasks were the test collection used in TSC-2, which include 30 document sets (224 documents). After the range of each parameter was set manually, the system changed the values of the parameters within the range and iterated summarization on the training data. Each score was recorded whenever the parameter values were changed, and the parameter values having the best score were stored.

2.3 System with categorization of feature values (SysC)

This system uses categorized feature values so that features can be combined discretely. We used decision trees for combining these feature values. For generating decision trees, we used C4.5 package [11].

The types of features used for generating decision trees are the same as SysL, while each function in a feature is separated. For example, three functions in a sentence-position feature are used for generating decision trees as if they are different features. A value of feature is categorized by 0.1. Since all feature values are on a scale of zero to one, 11 binary features are generated from one feature value ($[0, 0.1), [0.1, 0.2), \ldots, [0.9, 1.0]$).

We used the system SysC for generating both abstracts and extracts for the formal run. We also submitted results of extracts using SysL.

2.4 Similarity between sentences

Our summarization system uses a module to estimate the similarity between sentences. Similarity values are used to either select one key sentences among semantically similar sentences or output a set of similar sentences with high sentence scores.

There are two kinds of similarity functions to check if a given sentence is redundant or necessary. The assumption here is:

1. If two sentences are similar and have no NEs:
   the sentence pair has the same contents.
2. If two sentences are similar and share NEs:
   the sentence pair has the same contents.
3. If two sentences are similar and both have different NE tokens of the same types:
   the sentence pair has the similar structure but different contents.

For example, in articles about one criminal case the preceding facts of the case are described repeatedly. These repetition should be removed. On the other hand, when each article describes an earthquake that occurred at different place in a given document set, expressions in the articles are typical and similar, but tell us different information. These expression should be included in the summary of the document set. Based on that assumption, our system calculates two similarity functions; one is based on content words (SimW), and the other similarity function is based on NEs (SimN).

The system uses Dice, Jaccard, or cosine coefficients as a similarity measure based on the number of words between two sentences. When two sentences are represented as word vectors S_x and S_y, each coefficient between them is calculated as follows[7]:

Table 1. Evaluation results of the extraction task

	Short		Long	
Sys.	Cov.	Prec.	Cov.	Prec.
SysC	0.222	0.314	0.313	0.432
SysL	0.293	0.378	0.295	0.416
LEAD	0.212	0.426	0.259	0.539

Table 2. Evaluation results of the abstraction task

	Content evaluation		Pseudo-question answering			
			Exact		Edit	
Sys.	Short	Long	Short	Long	Short	Long
SysC	0.188	0.240	0.257	0.266	0.556	0.602
LEAD	0.160	0.159	0.300	0.275	0.589	0.602

$$\mathit{Dice}(S_x, S_y) = \frac{2|S_x \cap S_y|}{|S_x| + |S_y|}$$
$$\mathit{Jaccard}(S_x, S_y) = \frac{|S_x \cap S_y|}{|S_x \cup S_y|}$$
$$\mathit{cosine}(S_x, S_y) = \frac{|S_x \cap S_y|}{\sqrt{|S_x| \times |S_y|}}$$

The following three types of weights at each word can also be selected from the following:

Binary: if the word appears on the sentence, the weight is set to 1. The weight is set to 0 otherwise.
Tf: the term frequency of the word
Tf*idf: the tf*idf value of the word

The system uses one of coefficients with one of the weights to calculate similarities. For calculating both values of SimW and SimN, we used cosine coefficient with the binary weight from results at the training stage.

Our system assumes that a given sentence pair is similar each other when the coefficient between the sentences is higher than a threshold, i.e. two sentences are regarded as similar when SimW is greater than T_{sw}, and regarded as sharing NEs when SimN is greater than T_{sn}. The values of both thresholds for T_{sw} and T_{sn} were set to 0.4 at the training stage.

3 Evaluation results

In this section, we show evaluation results of our system for the TSC-3 tasks. Table 1 shows evaluation results of the extraction task. Both of our systems (SysL and SysC) had better coverage than the Lead-based method, but the values of the precision were worse. In the comparison between SysL and SysC, SysL had better results at short-length extracts, and SysC had better results at long-length extracts.

Table 2 shows evaluation results of the abstraction task. As for content evaluation, our system's performance was better than those of the Lead-based method. As for evaluation with pseudo-question answering, our system's performance was not so good compared to those of the Lead-based method. One of possible reasons the performance degraded is that decision trees from training data fell into overtraining. Table 3 shows results on readability evaluation of abstracts with quality questions. Our system has better results than the Lead-based system on q00, q01, q04, q07, q10, q14 and q15. Especially the result on q00 was better than that of any other system.

Table 4 shows that evaluation results of extraction tasks when one feature was discarded from feature combinations. We introduced here another summarization system (SysM) which uses a maximum entropy (ME) framework [2, 3, 1]. The model is represented by Eq. (1):

$$p_\lambda(a|b) = \frac{\exp\left(\sum_{i,j} \lambda_{i,j} g_{i,j}(a,b)\right)}{Z_\lambda(b)}$$
$$Z_\lambda(b) = \sum_a \exp\left(\sum_{i,j} \lambda_{i,j} g_{i,j}(a,b)\right),$$

where a is one of categories for classification (This is called a "future."). The value of the future is 1 (key sentence) or 0 (otherwise) here to indicate results of sentence extraction. b is conditioning information that enables us to make a decision among the space of futures (This is called a "history."), and $Z_\lambda(b)$ is a normalizing constant determined by the requirement that $\sum_a p_\lambda(a|b) = 1$ for all b. The computation of $p_\lambda(a|b)$ in any ME model is dependent on a set of "features" which are binary functions of the history and future. For instance, one of our features is

$$g_{i,j}(a,b) = \begin{cases} 1 : \text{if } has(b, f_j) = 1 \ \& \ a = a_i \\ 0 : \text{otherwise.} \end{cases}$$

Here "$has(b, f_j)$" is a binary function that returns 1 if the history b has feature f_j. The features used in our experiments are the same as used for generating decision trees.

The comparison among feature combinations shows that the 'sentence-length' (L) feature was the most effective feature to the coverage at short-length extracts among all systems. At long-length extracts, the 'sentence-position' (P) and 'question' (Q) features had also some effects to the results besides the sentence-length. Precision scores for SysM indicated that the 'headline' (H) feature was not effective, because the precision was better when the feature was not used. The 'tf*idf' (F) feature for SysC at short-length extracts had also a harmful effect to the performance. Comparing systems with all features, while

Table 3. Results on readability evaluation of abstracts

	Short		Long		Total	
QQ	Sys.	LEAD	Sys.	LEAD	Sys.	LEAD
q00	0.033	1.500	0.300	2.833	0.167	2.167
q01	0.567	1.267	0.500	2.100	0.533	1.683
q02	0.700	0.267	2.100	0.633	1.400	0.450
q03	0.667	0.267	0.333	0.367	0.500	0.317
q04	1.567	1.667	2.667	4.300	2.117	2.983
q05	1.400	0.067	3.600	0.300	2.500	0.183
q06	0.500	0.767	1.500	1.033	1.000	0.900
q07	0.267	1.533	0.467	4.333	0.367	2.933
q08	-0.500	0.267	-0.900	-0.333	-0.700	-0.033
q09	0.100	0.067	0.133	0.067	0.117	0.067
q10	0.367	1.667	0.233	5.133	0.300	3.400
q11	0.033	0.000	0.000	0.000	0.017	0.000
q12	0.000	0.000	0.067	0.000	0.033	0.000
q13	0.033	0.033	0.000	0.000	0.017	0.017
q14	0.000	0.033	0.033	0.067	0.017	0.050
q15	0.100	0.200	0.233	0.433	0.167	0.317

Table 4. Comparison of evaluation results among systems and feature combinations (Extraction)

	Short						Long					
	SysL		SysC		SysM		SysL		SysC		SysM	
Features	Cov.	Pre.	Cov.	Pre.	Cov.	Pre.	Cov.	Pre.	Cov.	Pre.	Cov.	Pre.
-LFHTQ	0.254	0.328	0.224	0.324	0.267	0.344	0.276	0.398	0.294	0.430	0.281	0.409
P-FHTQ	0.242	0.340	0.218	0.324	0.260	0.373	0.281	0.410	0.269	0.418	0.294	0.423
PL-HTQ	0.280	0.354	0.224	0.326	0.271	0.350	0.287	0.410	0.277	0.411	0.328	0.464
PLF-TQ	0.260	0.358	0.231	0.347	0.283	0.379	0.285	0.406	0.277	0.417	0.307	0.451
PLFH-Q	0.293	0.378	0.222	0.314	0.272	0.338	0.289	0.413	0.313	0.432	0.313	0.435
PLFHT-	0.259	0.337	0.222	0.338	0.280	0.338	0.278	0.408	0.293	0.428	0.311	0.429
ALL	0.293	0.378	0.222	0.314	0.281	0.376	0.295	0.416	0.313	0.432	0.309	0.443

SysL was the best at short-length extracts, we can see that SysC and SysM had better performance than SysL at long-length extracts.

4 Concluding remarks

We have presented two types of summarization systems and the results at the TSC-3 tasks. One system is to use linear combination of features to calculate the total score of a sentence, the other is to categorize feature values and combine feature categories to extract key sentences by generating decision trees. Comparison of results of two systems at the extraction task showed that the system with linear combination of features had better results at short-length extracts, and the system using combined feature categories had better results at long-length extracts. Results of quality questions on the abstraction task indicated that redundancy detection used in our system was effective to remove redundant sentences in abstracts. As future work, we consider the use of other machine-learning modules to combine features instead of a decision tree.

References

[1] A. L. Berger, S. A. D. Pietra, and V. J. D. Pietra. A Maximum Entropy Approach to Natural Language Processing. *Computational Linguistics*, 22(1):39–71, 1996.

[2] E. T. Jaynes. Information Theory and Statistical Mechanics. *Physical Review*, 106:620–630, 1957.

[3] E. T. Jaynes. Where do we Stand on Maximum Entropy? In R. D. Levine and M. Tribus, editors, *The Maximum Entropy Formalism*. M. I. T. Press, 1979.

[4] S. Kurohashi and M. Nagao. *Japanese Morphological Analyzing System: JUMAN version 3.61*. Kyoto University, 1999.

[5] Madani. http://classes.seattleu.edu/computer_science/csse470/Madani/ABCs.html. ABCs of Text Categorization.

[6] I. Mani. *Automatic Summarization*. John Benjamins Publishing Company, 2001.

[7] C. D. Manning and H. Schuetze. *Foundations of Statistical Natural Language Processing*. The MIT Press, 2000.

[8] NII, editor. *Proceedings of the Third NTCIR Workshop on Research in Information Retrieval, Automatic Text Summarization and Question Answering*, Tokyo, Japan, October 2002. National Institute of Informatics.

[9] C. Nobata, S. Sekine, and J. Tsujii. Analysis on difficulty indices for japanese named entity task. *Journal of Natural Language Processing (in Japanese)*, 10(1), 2003.

[10] M. Okumura and H. Nanba. Automated Text Summarization: A Survey (in Japanese). *Journal of Natural Language Processing*, 6(6):1–26, 1999.

[11] J. R. Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann Publishers, Inc., San Mateo, California, 1993.

[12] S. Sekine, K. Sudo, and C. Nobata. Extended Named Entity Hierarchy. In *Proceedings of the LREC-2002 Conference*, pages 1818–1824, 2002.