Tsinghua University at the summarization track of TAC 2008

*Shouyuan Chen, *Yuanming Yu, *Chong Long,

Feng Jin, Lijing Qin, Minlie Huang, Xiaoyan Zhu[1,2,3]

[1]Dept. of Computer Science and Technology,
[2]Tsinghua National Laboratory for Information Science and Technology,
[3]State Key Laboratory on Intelligent Technology and Systems,
Tsinghua University, Beijing 100084, China

*The three authors should be all first-authors.

Corresponding author: Minlie Huang {aihuang@tsinghua.edu.cn}

Abstract

This paper presents our extractive summarization systems at the update summarization track of TAC 2008. We proposed two novel methods, one is based on the information distance theory, and the other is based on the sentence centrality which derives from the centrality concept in the graph theory. The evaluation results show that the two submitted runs are very competitive to generate extractive summaries.

1 Introduction

We participated in the update summarization track of TAC 2008. The update summarization task is to write a short (~ 100 words) summary of a set of newswire articles, under the assumption that the user has already read a given set of earlier articles. The summaries will be evaluated for readability and content (based on Columbia University's Pyramid Method) [1].

We developed two systems for the task. The first one was based on the Kolmogorov complexity and information distance theory. Here we provide an initial pragmatic study of measuring information shared by many objects, which can be applied to make a summary given a group of news articles: firstly, the optimal summary can be viewed as the one with the smallest information distance to all original news articles. In other words, this summary can cover the most information contents shared by all those articles. And then, the text summarization problem is converted into an optimization problem limited by the summary's information content (the length of summary). Finally, to solve this optimization problem, we proposed an approach to approximate *K(.)* and *D(.,.)*. The results ranked 1st by average overall responsiveness, 3rd by average linguistic quality, 5th by average modified pyramid score, and 5th by macroaverage modified score.

The second system was based on the centrality concepts within the graph theory. We constructed a graph model in which nodes refer to sentences and edges represent the similarities between sentences which are calculated by a latent semantic indexing (LSI) algorithm. We proposed a variation of eigenvector centrality calculation method on the graph model to estimate the relevant importance of sentences, which contributed to choose the candidate sentences and score the sentences as a feature. In addition, the background information provided in previously seen documents was reduced by performing the eigenvector calculation on a combination of two graph models of currently processing documents and previously seen documents. In short, the centrality of a sentence would be increased by the similarities with important sentences in current documents, while it had a negative correlation with major information presented in former documents set. The results ranked 1st by average overall responsiveness, 2nd by average linguistic quality, 7th by average modified pyramid score, and 7th by macroaverage modified score.

In the following sections, we introduce the two systems respectively, and finally concluding remarks are made.

2 The first system: information-distance based update summarization

2.1 Kolmogorov complexity and information distance

Kolmogorov complexity was introduced almost half a century ago by R. Solomonoff, A.N. Kolmogorov and G. Chaitin, see [2]. It is now widely accepted as an information theory for individual objects parallel to that of Shannon's information theory which is defined on an ensemble of objects. Fix a universal Turing machine *U*, the Kolmogorov complexity of a binary string x condition to another binary string y, $K_U(x|y)$, is the length of the shortest (prefix-free) program for U that outputs x with input y. It can be shown that for different universal Turing machine Us, for all x,y

$$K_U(x|y)=K_{U'}(x|y)+C,$$

where the constant C depends only on U'. Thus $K_U(x|y)$ can be simply written as $K(x|y)$. They write $K_U(x|y)$, where ε is the empty string, as $K(x)$. For a comprehensive study of Kolmogorov complexity and its applications, see [3].

Given Kolmogorov complexity, information distance was defined in [2] as the energy to convert between x and y to be the smallest number of bits needed to convert from x to y and vice versa. That is, with respect to a universal Turing machine U, the cost of conversion between x and y is:

$$E(x,y)=\min\{|p|:U(x,p)=y, U(y,p)=x\}.$$

It is clear that $E(x,y)\leq K(x|y)+K(y|x)$. From this observation, and some other concerns, Bennett et al. have defined the sum distance in [3]:

$$D_{sum}(x,y)=K(x|y)+K(y|x).$$

However, the following theorem proved in [3] was a surprise:

$$E(x,y)=\max\{K(x|y),K(y|x)\}.$$

Thus, the max distance was defined in [3]:

$$D_{max}(x,y)=\max\{K(x|y),K(y|x)\} \quad (2.1)$$

Now given an object x, we can get its

information content through its Kolmogorov complexity K(x); given two objects K(x) and K(y), we can get the information between them through $D_{max}(x,y)$. However, given n objects $x_1,x_2,\ldots,x_n$, how to get the information shared by them?

If only the most shared information, which can be viewed as "information center" c, is needed, we can find the one with the one with minimal $D_{max}(c, x_1,x_2,\ldots,x_n)$.

2.2 Application in Text Summarization

TAC update summarization task is to write a short summary S of n newswire articles $B_1,B_2,\ldots,B_n$, under the assumption that the user has already read a given set of earlier m articles $A_1,A_2,\ldots,A_m$. According to the theory of our theory, S must satisfy the following condition:

$$minD_{max}(S, B_1B_2\ldots B_n| A_1A_2\ldots A_m),\ |S|\le\Theta$$

Since the information distance between the summary and these articles is not computable, in the following section we will take some steps to approximate it. Although some steps can be regarded as a very rough estimation of information content (or Kolmogorov complexity) of the review articles, from the experiment results in section 3 we will see that this already gives very good practical results.

2.3 How to approximate $D_{max}(.,.)$

Let $A\cup B$ to be all information contained in article A or B, and A\B to be all information contains in A but not in B, then we can get:

$$K(AB)=K(A\cup B) \qquad K(A|B)=K(A\backslash B)$$

Suppose S contains l sentences $S_1,S_2,\ldots,S_l$, moreover, to be an ideal summary, all the information contained in S should also be in $B_1,B_2,\ldots,B_n$, but not in $A_1,A_2,\ldots,A_m$, that is

$$S_1,S_2,\ldots,S_l\in B_1B_2\ldots B_n\backslash A_1,A_2,\ldots,A_m$$

According to (2.1):

$$D_{max}(S, B_1B_2\ldots B_n| A_1A_2\ldots A_m)$$
$$=\max\{K(S|B_1B_2\ldots B_nA_1A_2\ldots A_m),$$
$$K(B_1B_2\ldots B_n| SA_1A_2\ldots A_m)\}$$

Since $S_1,S_2,\ldots,S_l\in B_1B_2\ldots B_n\backslash A_1,A_2,\ldots,A_m$, $K(S|B_1B_2\ldots B_nA_1A_2\ldots A_m)\le K(B_1B_2\ldots B_n| SA_1A_2\ldots A_m)\}$, therefore,

$$D_{max}(S, B_1B_2\ldots B_n| A_1A_2\ldots A_m)$$
$$=K(B_1B_2\ldots B_n| S A_1A_2\ldots A_m)$$
$$=K(B_1B_2\ldots B_n| S_1S_2\ldots S_lA_1A_2\ldots A_m)$$
$$=K(B_1B_2\ldots B_n\backslash A_1A_2\ldots A_m)\backslash S_1S_2\ldots S_l)$$

As $B_1B_2\ldots B_n$ and $A_1A_2\ldots A_m$ are already known and $S_1S_2\ldots S_l$ are in $B_1B_2\ldots B_n\backslash A_1A_2\ldots A_m$, the more information $S_1S_2\ldots S_l$ contains, the less one $(B_1B_2\ldots B_n\backslash A_1A_2\ldots A_m)\backslash S_1S_2\ldots S_l$ has. Therefore, in order to get $minD_{max}$, we need to get $max\{K(S_1S_2\ldots S_l)\}$.

Suppose in $S_1,S_2,\ldots,S_l$, each "important" word carries one unit of information, then the Kolmogorov complexity can be intuitively estimated by

$$K(S) =|S|$$

However, in a sentence, which words are "important"? In our method, two conditions must be satisfied: (1) they are not "stop-words"; (2) their document frequency (df) is larger than a threshold. Intuitively, those "not important" words contribute much less information to the whole sentence so they can be ignored in our approximation.

2.4 Sentence selection

The next problem is how to generate $S_1S_2\ldots S_l$ to make a good summary S.

It is clear that under most conditions, in an article, there is always one sentence which can cover most of this article's information. Therefore, for one article, we tend to only find one sentence to represent the whole article under the given topic. According to the task requirement, the most representative must be completely related to the topic.

In TAC 2008, for a topic, named entities, such as special words or phrases of a person, a place, an organization, et al., contain the most part of the information, and they are “key words”. For example, in the following two topics’ narratives:

(1) Describe developments in the production and launch of the **Airbus A380.**

(2) Describe steps taken and worldwide reaction prior to introduction of the **Euro** on **January 1, 1999**. Include predictions and expectations reported in the press.

Sentence (1) has two key words “Airbus” and “A380”, and (2) has “Euro” and “January 1, 1999”. Moreover, “January 1, 1999” will have the different forms, such as “January 1st, 1999” and “1/1/1999”. A most representative sentence of an article should contain the information similar to a topic, which means it must have similar key words.

Here named entity recognition method [3] is used to all useful entities contained in a topic’s title, narrative, and each sentence of an article. We also develop a method to select the words or phrases that indicates the date. In an article B, the sentence with minimal distance to the union of its topic’s title and narrative (named *T*) is just the most representative one:

$$\min D_{max}(T,P) \quad P \in B$$

Similar to estimating $K(S_1S_2...S_l)$ mentioned in the above section: $K(S) = |S|$. Only one different is that “important words” refer to those have been annotated as name or date entities which named key words.

With this method, for each document B_i ($1 \leq i \leq n$) in cluster B, one representative sentence S_i can be found, so there should be n representative sentences. However, under certain situations, two or more articles’ representative sentences (written as S_i and S_j) may have nearly the same meaning, which will contribute less to $K(S_iS_j)$ due to much overlap of S_i and S_j. Therefore, if two sentences S_i and S_j have (1) more than eight continuous common words, or (2) more than 60% common words, the sentence with smallest distance to the topic will be chosen from these two articles (except S_i and S_j**)** to be replaced with one of them.

Finally, using these methods mentioned above, the combination $S_1S_2...S_l$ with most $K(S_1S_2...S_l)$ is selected into the final summary S.

2.5 Experiment Results

Our text summarization method founded on Kolmogorov complexity is complemented and tested in TAC 2008. The experiment results under pyramid evaluation methods are as follows:

Evaluation Method	Best Result	Our Result	Rank
Average Modified Score	0.336	0.309	5
Macroaverage Modified Score with 3 models	0.331	0.304	5
Average Linguistic Quality	3.333	2.958	3
Average Overall Responsiveness	2.667	2.667	1

Table 1. The evaluation results for the first system.

Obviously, our method has got generally higher rankings under the pyramid methods, however, except average overall responsiveness evaluation, the best results are significantly better than ours.

3 The second system: sentence centrality based update summarization

The system works as follows: First, we

stemmed all the words and segmented articles into sentences (viewed as documents hereafter). Then we built a matrix W_{t*d} (terms *sentences) according to the term frequency and document frequency, and soon later constructed a sentence-sentence similarity matrix *SIM* by employing an LSI algorithm to calculate the similarities between sentences.

After that, we propose an approach to calculate sentence centrality on the *SIM* matrix. Informally, the sentence centrality is the measure of the relevant importance of a sentence in the current processing cluster, and its importance is decreased if it was previously seen in given clusters. All these centralities, which contribute to choose the candidate sentences, are computed mainly based on an eigenvector calculation.

As the last step of our summarization process, we generate the summary by searching the combinations of sentences with a maximal summary score in the candidate set. In this process, we score the summary as a whole by combining the word scores, sentence centralities. And the redundancy among sentences is also considered in the summary scoring schema.

We tuned the weights of our method manually based on the maximization of the ROUGE-2 measure upon the DUC 2007 Update Task corpus.

3.1 Scoring terms

We design a scoring function to measure how important each word is. Basically, the term frequency and document frequency are considered to score a term. We set a weight according to the position where the term is observed, which are shown in Table 2.

Position	Weight
Narrative&Topic	8
Headline	4
[a]First Sentence of a document	1
[b]First Sentence of a paragraph	0.2
Not a nor b	0.2
others	0.04

Table 2. Weights for the term frequency

Then we design the following function to score a term:

$$\boldsymbol{Score(w)=tf(w)^{0.4}*F_d(df(w))} \qquad (3.1)$$

Where the *tf(w)* is the total frequency of term *w* across all documents, *df(w)* is the document frequency, and *Fd(.)* is a weight function as designed in the below table:

df(w)	*Fd(.)*	*df(w)*	*Fd(.)*
0	0	6	8
1	1	7	8.2
2	2	8	8.4
3	4	9	8.5
4	6	10	8.6
5	7	>10	8.6

Table 3. Weights for the document frequency

Using the scoring function, we only keep the top 300 terms in this task to build the *W* matrix and *SIM* matrix.

3.2 Building the *W* matrix and *SIM* matrix

After the 300 terms selected out, the documents are segmented into sentences. Then we build the $W=(w_{ij})_{t\times d}$ matrix (term-document matrix), where w_{ij} is the frequency of the *i*-th word appearing in the *j*-th sentence. *t* is the number of words, and *d* is the number of sentences.

Then we use LSI method [5] to decompose the *W* matrix into the product of three matrixes:

$$\mathbf{(w)}_{t\times d}\approx\mathbf{(T)}_{t\times k}\,\mathbf{(S)}_{k\times k}\,\mathbf{(D^T)}_{k\times d} \qquad (3.2)$$

where *k* is less than the rank of matrix W. Define $Y=X^TX=DS^2D^T$, and the $SIM=(Sim_{i,j})$

matrix can be defined as follows:

$$Sim_{i,j} = \frac{Y_{ij}}{\sqrt{Y_{ii}}\sqrt{Y_{jj}}} \qquad (3.3)$$

For a single cluster, Sim_{ij} denotes the similarity between sentence *i* and sentence *j*. If we generate a summary for cluster A, the matrix can be used to construct the graph. Remember that the update summarization task requires providing as much new information as possible for cluster B if we have already read cluster A. Let us go back to the problem: given cluster A, we need to generate a summary for cluster B, to provide as much new information as possible.

Suppose there are n_A sentences in cluster *A*, and n_B in cluster B, then the original *W* matrix could be rewritten as W'=(W_A W_B) where W_A is the term-document matrix for cluster *A* and W_B for cluster *B*. The rows of *W'* represents important words selected from both cluster A and B. Following the line of thinking, the similarity matrix can be formulated as follows:

$$SIM = \begin{pmatrix} SIM_{AA} & SIM_{AB} \\ SIM_{BA} & SIM_{BB} \end{pmatrix} \qquad (3.4)$$

where the SIM_{AA} is the similarity matrix for sentences within cluster A and SIM_{AB} is the similarity matrix between sentences in cluster A and those in cluster B.

3.3 Computing the sentence centrality

The centrality is a concept of graph theory for measuring how important a vertex is in a graph. The idea is similar to the process of extractive summarization which is to find important representative sentences from documents. In our method, each sentence is viewed as a vertex, and there is an edge connecting sentence *i* and sentence *j* with weight Sim_{ij} if i≠j. On such a graph, we use a power iteration algorithm to compute the centrality of a vertex.

The centrality idea is that a sentence is important if it is similar to other important sentences. Translate this to the graph language: if a vertex connects with other important vertices with high weights, the vertex itself is important. Suppose the centrality of sentence *s* is C_s, then the idea is formally as:

$$\lambda C_s = \sum_{r \in D, r \neq s} Sim_{r,s} C_r \qquad (3.5)$$

where $Sim_{r,s}$ is the similarity between sentence *r* and *s*, and $D=\{1,2,\ldots,d\}$ (*d*: the number of sentences). Define the adjacency matrix $M=(m_{ij})$ as follows:

$$m_{ij} = \begin{cases} Sim_{ij}, i \neq j \\ 0, i = j \end{cases}$$

Let $C=(C_1,C_2,\ldots,C_d)^T$, we have the matrix form for the formula (3.5):

$$\lambda \boldsymbol{C} = \boldsymbol{M} * \boldsymbol{C} \qquad (3.6)$$

Notice that C is the eigenvector of M, and λ is the corresponding eigenvalue. This inspires us to apply the classic eigenvector calculation algorithm. In our system, we use the power iteration algorithm.

(1) Choose an arbitrary random vector $U=(u_1,u_2,\ldots,u_d)^T$ that satisfies $\max_{1\le i\le d} |u_i| \le 1$;

(2) $V=M*U$, $k = \max_{1\le i\le d}\{|v_i|\}$;

(3) Let $u' = v/k$, $\delta = \max_{1\le i\le d} |u_i - u_i|$;

(4) If δ<0.001 exit the iteration; otherwise go to step 2.

Note that *M* is a real symmetric matrix and hence its eigenvalues should be real numbers. It can be shown that when the eigenvalues of *M*, λ_1, λ_2,λ_3...λ_d, satisfy $\lambda_1>\lambda_2>\lambda_3>\ldots>\lambda_d$, this algorithm will terminate in finite steps and returns the largest eigenvalue and its corresponding eigenvector.

From the viewpoint of graph theory, vector C is actually the "eigenvector centrality", which represents the importance of vertices in a graph whose adjacency matrix is M.

Our approach is similar to LexRank [6] and TextRank [7], but we developed a completely different way to compute the sentence centrality for the update summarization task.

3.4 Computing the sentence centrality for cluster B with cluster A given

In cluster B, the system already has "read" documents in cluster A. Thus those sentences already highlighted in cluster A should not be emphasized too much now in cluster B. The system should provide more new important information without mentions in already seen documents. Intuitively, if a sentence in cluster B is very similar to an important sentence in cluster A, the sentence should not be included in the summary of cluster B since the information is already covered by cluster A. Hence, we should take into account the similarity between sentences in cluster B and sentences in cluster A. We change the equation as follows:

$$\lambda C_s = \left[\left(\sum_{r \in D_B, r \neq s} Sim_{r,s} C_r \right) - \beta \left(\sum_{r \in D_A, r \neq s} Sim_{r,s} C_r \right) \right] \quad (3.7)$$

Where β is a penalty constant which is set to 0.8 in practice.

Noting the way that we construct the SIM matrix for cluster B as shown in (3.4), SIM can be written as following:

$$SIM' = \begin{pmatrix} SIM_{AA} & -\beta SIM_{AB} \\ -\beta SIM_{BA} & SIM_{BB} \end{pmatrix} \quad (3.8)$$

where SIM_{AA} has $|D_A|$ rows, SIM_{BB} has $|D_B|$ rows.

Apply the same power iteration algorithm on this matrix, we can obtain the centralities of sentences and furthermore we can select out those important sentences as candidate sentences of the final summary.

3.5 Summary extraction

The extraction process has three steps. First, select candidate sentences according to the centrality computed in section 3.4. The candidates consist of the following two parts: (1) the first sentence of each article whose sentence centrality is among the top 1/3 of all sentences in the cluster; (2) sentences that are not the first sentence, but with high centrality scores. The size of the candidate sentence set is set to 15 in our system.

Second, search for the best sentence combination. We enumerate all possible sentence combinations from the candidate set, and score the summary as a whole that is different from the traditional approaches which scores sentences independently. Formally, the algorithm can be written as the following:

(1) *bestScore*=-∞, *bestSum*=Θ;
(2) *Find a combination* $S=s_1,s_2,\ldots,s_m$ *where* s_i *is chosen from the candidate set, and the total number of words in S satisfies the length limit (100 in this task);*
(3) Calculate the score of the summary, SumScore(S), which will be introduced soon later. If SumScore(S)>bestScore, bestScore=SumScore(S), bestSum=S;
(4) If all combinations are enumerated, exit; otherwise go to step (2).

Now it is the turn to describe the scoring function of a summary. Our consideration is that a good summary should cover various kinds of important information mentioned in the articles, and it should have as little redundant information as possible.

Firstly, we calculate the maximal similarity (which could be treated as a measurement of redundancy) between sentences in the summary *S*, as follows:

$$R(S) = \max_{1 \le i,j \le m} Sim_{s_i,s_j}$$

We should not accept the summary whose information is highly redundant. In our system, if *R(S)>0.5*, the summary is discarded.

Secondly, for each word w, observed in S, we evaluate its contribution to the summary score as following:

$$count_w = \sum_{1 \le i \le m, w \in s_i} occur_w(s_i) C_{s_i}^{0.5}$$

where $occur_w(s_i)$ represents the occurrence number of word w in sentence s_i, C_{si} is the sentence centrality of s_i which is described in section 3.4.

We are now ready to combine the word score *Score(w)* shown in the formula (3.1) and design the final score of the summary. We define $maxscore = max_{any\ w}\ \{Score(w)\}$, *power(w)*, and finally *SumScore(S)* as below:

$$Power(w) = \left(1 - 0.45\left(\frac{Score(w)}{maxscore}\right)^{0.15}\right) / 2$$

$$SumScore(S) = \sum_{w \in S} (count_w)^{Power(w)}$$

The final step of summary extraction is to re-order sentences within the extracted summary. Suppose the extracted summary is $bestSum = \{b_1, b_2, ..., b_n\}$, then we compute the summary score of each individual sentence b_i as follows:

$$P(b_i) = SumScore(\{b_i\}),\ 1 \le i \le n$$

Sort these sentences according to $P(b_i)$ in an descending order, and this is the final summary.

3.6 Evaluation results

The evaluation results from the official evaluation from TAC 2008 are shown in the following table. Among 58 submitted runs, our results are competitive.

Evaluation Method	Best Result	Our Result	Rank
Average Modified Score	0.336	0.304	7
Macroaverage Modified Score with 3 models	0.331	0.299	7
Average Linguistic Quality	3.333	3.073	2
Average Overall Responsiveness	2.667	2.667	1

Table 4. The Evaluation results for the second system.

Conclusion

In this paper, we have presented two systems for the update summarization task of TAC 2008. The first system is based on the information distance theory, and the other is based the graph centrality. Although there are still many parameters to be tuned experimentally, the two methods have already contributed very competitive results during the official evaluation.

Still, fine tuning of these two systems, and more sophisticated natural language processing techniques, are awaiting to be exploited.

Acknowledgement

This work was supported by National Natural Science Foundation of China (60572084, 60621062, 60803075), National Basic Research Program of China (2007CB311003), and 985 key projects (SIST3002), as well as by Tsinghua Basic Research Foundation under grants no. 052220205 and no. 053220002.

References

[1] http://www.nist.gov/tac/tracks/2008/summarization/index.html

[2] M. Li and P. Vit′anyi. An Introduction to Kolmogorov Complexity and its Applications (2nd Edition). Springer-Verlag, 1997.

[3] C. Bennett, P. Gacs, M. Li, P. Vit′anyi, and W. Zurek. Information distance. IEEE Transactions on Information Theory, 44(4):1407–1423, July 1998.

[4] Vijay Krishnan and Christopher D. Manning. An Effective Two-Stage Model for Exploiting Non-Local Dependencies in Named Entity Recognition. In Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics. 2006

[5] Scott Deerwester, Susan T. Dumais, George W. Furnas, Thomas K. Landauer, and Richard Harshman. Indexing by Latent Semantic Analysis. Journal of the American Society of Information Science, Vol. 41, No. 6. (1990), pp. 391-407.

[6] Gűneş, Erkan and Dragomir R. Radev, LexRank: Graph-based Centrality as Salience in Text Summarization. Journal of Artificial Intelligence Research (JAIR), Vol. 22 (2004), pp. 457-479.

[7] Rada Mihalcea and Paul Tarau, TextRank: Bringing Order into Texts. In Proceedings of EMNLP 2004.