News Aggregating System with Automatic Summarization Based on Local Multiple Alignment

Takaharu Takeda* Atsuhiro Takasu**
Graduate University for Advanced Studies * National Institute of Informatics **
2-1-2 Hitotsubashi, Chiyoda, Tokyo Japan
{takeda_takaharu* , takasu**}@nii.ac.jp

Abstract

This paper proposes a news article aggregation system with an automatic summarization function. It provides integrated and effective access to news articles from various news sites and presents a brief summary of them. The system consists of a collector, a topic detector, and a summarizer for the news articles. This paper particularly focuses on the system's efficient summarization technique for handling large amounts of crawled news articles.

1. Introduction

Various kinds of information sources are currently accessible through the Internet. News sites are one of the most useful sources of information from among them. However, users typically want to read only those articles that they are interested in. Therefore, systems need to assist users select news according to their personal preferences. A new technology that can integrate dynamically changed documents is needed, because these documents are distributed and frequently updated. In addition, such a system should provide various functions.

The categorization of news articles is the first step towards achieving this goal. Many news sites provide articles by listing them according to their categories, such as politics and sports. However, the granularities of the categories are too coarse to filter through all the articles. Topic detection and tracking technology [5] detects the events described in the news articles and makes clusters of articles according to these events. With this technology users can keep track of news events that interest them. Since the same event can be described from different aspects in different news articles, users often compare articles from different sources. Therefore, the system needs to be able to gather news articles from various sources and link the articles describing the same topic to each other.

Topics usually consist of multiple news articles. Therefore, users need to read some of them to understand the contents of the topic. A concise description or summary of each topic helps the user to better understand what the topic is about. Document summarization technology is used to automatically make concise descriptions from news clusters. Topic detection and document summarization are the key technologies used to effectively provide a document stream, such as news articles, to users. This paper proposes a new document summarization method.

When utilizing these technologies, there are two aspects, i.e., the accuracy of the generated summaries and the processing efficiency, that are important. The main topic of the document summarization study is how to generate a good summary of documents and several methods that use deep natural language processing and machine learning techniques the researchers have proposed are described in the next section. However, these methods often require complicated and heavy computation. In the age of the WWW, we need to handle large number of documents, and we need a method that is efficient enough to process these documents in a stream on time without delay. We developed a news aggregation system that can efficiently handle a large amount of news with moderate summarization accuracy. We used multiple local alignment techniques for the document summarization in this system.

The rest of this paper is organized as follows. In Section 2, we briefly describe the related works focusing on the document summarization technique and approximate matching for the multiple document alignment. Section 3 presents an outline of the news aggregation system and Section 4 proposes an efficient document summarization method. Section 5 presents the experimental results concerning the accuracy and efficiency of the proposed method. Finally, Section 6 concludes this paper and addresses some future research directions.

2. RELATED WORK

Google News [1] is a typical news aggregating system. The news articles are frequently updated and their descriptions overlap each other. Users can more efficiently obtain the information they desire by digesting the news articles and removing the duplicate descriptions.

Text summarization is a key technique for digesting news articles and it has several approaches. Centroid-based summarization (CBS) [11-12] uses the centroids of the clusters of news articles produced by standard single-pass clustering systems (CIDR) [17] in order to extract sentences central to the topic. A centroid is a set of words that are statistically important to a cluster of documents,

and because of this, centroids can be used to classify relevant documents and to identify the salient sentences in a cluster.

The method described by R. Barzilay, K. McKeown, and M. Elhadad [14] generates a "concise summary" by identifying and synthesizing the similar elements across related texts from a set of documents. This system first determines how to combine propositions into a single sentence, and then it combines each set of propositions into a sentence, maps them from concepts to words, and builds a syntactic structure.

Maximal marginal relevance [6] is a widely used approach for information retrieval. It ranks documents according to a combined criterion of query relevance and information novelty within a document. It extracts the novel sentences and creates a summary from them.

The Columbia summarizer [15] uses an enhanced version of MultiGen [14], which integrates machine learning and statistical techniques to identify similar sentences across the inputted articles [16].

Our method generates a summary by extracting sentences directly from the original documents in the same way as proposed by J. Goldstein, V. Mittal, J. Carbonell, and M. Kantrowitz [13]. The difference between this study and theirs [13] is how we evaluate or select the sentences for the summary. Theirs generates a summary by extracting sentences that clearly represent a topic, whereas our method extracts sentences according to the frequency of specific word sequences.

Our method is similar to the Columbia summarizer in detecting similar sentences to summarize a text. However, ours can identify similar clauses and phrases as well as sentences even if the language does not have explicit word boundaries, as is the case with Chinese and Japanese. This is the most unique advantage of our method, and it is useful in a variety of ways.

Our method uses an approximate text search technique to detect similar sentences. A suffix tree [7] is made from all the suffixes of the given strings. Assuming the letter size is N, the tree is constructed in a calculation time less than O (N^2). In addition, a search is possible by sequentially tracing from the root of the tree. However, there is a weak point: the suffix tree requires a huge amount of memory.

Automaton is another technique for approximate searches. There are two types of automata that are used for approximate searches: non-deterministic and deterministic [8-9]. Both are robust against errors, both express errors as state transitions, and both can deal with any level of errors (such as erratum or typing errors).

Bit-parallelism [10] is a method for expressing one status of a deterministic automaton at one bit and simultaneously calculates multiple states in parallel. It can deal with varying levels of errors.

We would have to compare all the pairs of the portions of the text in order to apply the approximate search technique to a text summarization. This is computationally too expensive. Therefore, we developed an indexing method that compares any portion of a text in linear computation time to text length.

Our method depends on the idea [6, 11-16] of making an abstract by removing redundant descriptions that appear across all the documents. This problem can be defined using Local-Multiple Alignment [19] that has been often addressed in bioinformatics. We would introduce an idea to text-mining. We show that an approximate search without any linguistic knowledge is enough for detecting similar sentences. Our method can be applied to various languages because it does not use any specific language knowledge.

http://updatenews.sub.jp/

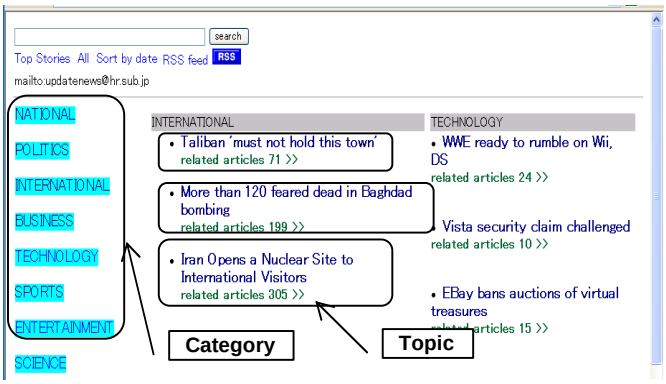

Figure 1: System overview

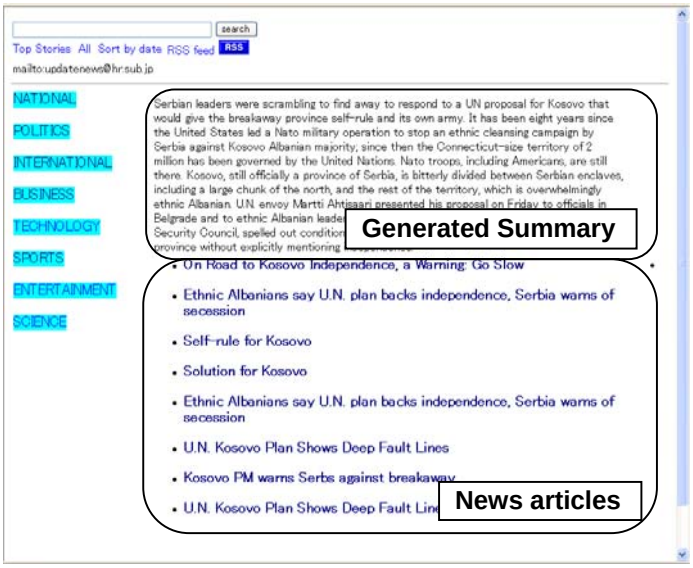

Figure 2: Example page for each topic

3. SYSTEM OVERVIEW

This section overviews our news aggregation system. Figure 1 shows the entry page of the system that gives an overview of the current news topics. As shown in the figure, news articles are currently classified into eight categories (National, Politics, International, Business, Technology, Sports, Entertainment, and Science) that are taken from the news sites. Each category consists of automatically detected topics that are listed with a headline and a number of articles.

When selecting a topic, the system gives an automatically generated summary of the topic with a list of news articles (Fig. 2). Users can read the article by clicking the headline of each news article. This system consists of three modules: a news collector, a topic detector, and a text summarizer.

3.1. NEWS COLLECTOR

The news collector gathers the news articles. It makes polling web sites to keep track of news updates and periodically obtains information concerning any uploaded news in an RSS format that contains the information about news updates. Fig. 3 shows an example of RSS format. For each updated news article marked with "item" tag, the news collector accesses the original news site and obtains the news content. We also use title and publication date in the RSS feed for managing the news articles.

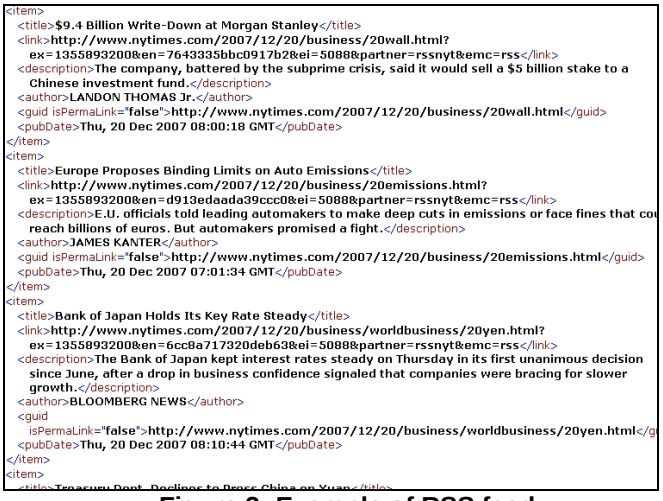

Figure 3: Example of RSS feed

It extracts the contents of news from the html source by removing tags and web advertisements using manually coded rules. In particular, it extracts strings inside the tags of specified ids, such as "main body" or "content", with an HTML parser.

3.2. TOPIC DETECTOR

Our system assigns a topic to the collected news articles. There are many studies being conducted on topic detection and tracking [1]. We use a simple topic detection method based on the vector space model. Each article is represented by a term vector (bag of words), and the value of each word is weighted by the inverse document frequency. Let $\{w_1,w_2,\ldots,w_n\}$ be a set of words used for representing the feature of the documents. The document frequency of a word w_i, denoted as $df(w_i)$, is the number of articles in the collection A_{all} containing the word w_i. Let $tf(w_i,a)$ denote the term frequency of the word w_i in an article a, i.e., the number of appearances of the word in the article. Then, each article a is represented by the following n-dimensional feature vector

$$\vec{a} = (f_1, f_2, \ldots, f_n) \ ,$$

where

$$f_i = tf(w_i, a) \times idf(w_i)$$
$$idf(w_i) = \log\left(\frac{|A_{all}|}{df(w_i)}\right) . \qquad (1)$$

Suppose there is a set $\{T_1,T_2,\ldots,T_m\}$ of topics where Ti denotes a set of articles included in the i-th topic. We then measure the similarity between a topic and a document by using a cosine measure that is frequently used in information retrieval. For a topic T and an article a, their similarity is measured as

$$sim(T, a) = \max_{b \in T} \frac{\vec{a} \cdot \vec{b}}{\|\vec{a}\| \times \|\vec{b}\|} , \qquad (2)$$

where $\|\vec{a}\|$ denotes the length of the vector. Then, the article a is classified into the most similar topic in terms of the similarity Eq. (2) if the similarity is more than a threshold. Otherwise, a is assigned to a new topic that contains only itself.

3.3. SUMMARIZER

The summarizer generates a summary of each topic from the documents assigned to a topic. The summarizer is described in Section 4.2.

4. NEWS ARTICLE SUMMARIZATION

The summarization is the dominant part of the news article processing. We developed an efficient summarization method using a light text processing. Documents belonging to the same topic often have the same information (such as paraphrases, translations, and even exactly the same letters). A simple multi-document summarization method (manual or automatic) omits repetitions and compiles the unique information [6]. This can be accomplished if the redundant regions are detected. Although such regions are easily detected if they are exactly the same, there are frequently subtle differences, such as insertions, deletions, and substitutions of alphabets or words. Therefore, we need an approximate matching that can handle such lexical and structural ambiguities to detect the redundant regions. The proposed method finds overlapping regions that can appear anywhere.

4.1. Similar Word Sequence

Our system uses a weighted edit distance [4] to measure the similarity between sentences (whole or parts of sentences), where the edit costs is determined based on the word

weight given by the inverse document frequencies (Eq. (1)). More precisely, for each word w in the given sentences, the cost of an insertion or deletion of w is defined as

$$c_i(w) = c_d(w) \equiv idf(w). \qquad (3)$$

This means we can remove or insert a word at a low cost if it appears frequently in the topic, which is similar to the case of stop words. By using the insert and delete operation costs, the substitution cost is defined as

$$c_s(w_1, w_2) \equiv \begin{cases} c_i(w_1) + c_d(w_2) & w_1 \neq w_2 \\ 0 & w_1 = w_2 \end{cases}. \qquad (4)$$

Table 1 shows an example of the insertion and deletion costs of words, i.e., the inverse document frequencies of the words. Significant words have higher values and meaningless words have lower ones.

We calculate the weighted edit distance of word sequences by using the DP-matching method, which is the same as that for the ordinary edit distance. First we denote some notations. Let us consider a word sequence $W=w_1w_2\ldots w_m$. We denote the i-the word of the sequence W as w_i. $W_{i:j}$ denotes the subsequence $w_iw_{i+1}\ldots w_j$. Note that $W_{1:k}$ denotes the prefix of W of length k. For a pair of word sequences W and Z, the distance of their prefixes is recurrently defined as

$$d(W_{1:i}, Z_{1:j}) \equiv \min \begin{cases} d(W_{1:i-1}, Z_{1:j-1}) + c_s(w_i, z_j) \\ d(W_{1:i-1}, Z_{1:j}) + c_d(w_i) \\ d(W_{1:i}, Z_{1:j-1}) + c_i(z_j) \end{cases}. \qquad (5)$$

For example, suppose the inverse document frequencies of words are given in Table 1. Let us consider the pair of word sequences "12 years as France's president" and "12 years as president of France" is 13. Then, the distances between any pair of their prefixes are given by Table 2.

For a pair W and Z of word sequences and a threshold σ, we call their prefixes $W_{1:l}$ and $Z_{1:m}$ are similar if $d(W_{1:l}, Z_{1:m}) < \sigma$ holds. For example, shaded cells in Table 2 show the similar prefixes when the threshold σ is 0.85. Let us define the length of the pair of prefixes $W_{1:l}$ and $Z_{1:m}$ as the shorter length of them, i.e.,

$$len(W_{1:l}, Z_{1:m}) \equiv \min\{l, m\}.$$

Then, we call the prefixes $W_{1:l}$ and $Z_{1:m}$ the longest similar prefixes if their length is longest among the similar pairs of prefixes. We denote the longest similar prefixes as W_* and Z_*. In Table 2, the longest similar prefixes are "12 years as France" and "12 years as president of France" and its length is 4. We want to enumerate the similar portions in sentences for summarization. We can enumerate them by enumerating the longest similar prefixes from pairs of suffixes of sentences.

w	12	years	as	France	President	of	's
c(w)	1.15	0.59	0.23	1.67	0.67	0.14	0.24

Table 1: Insertion and deletion costs of each word

	12	years	as	France	's	president
12	0	0.59	0.82	2.49	2.73	3.4
Years	0.59	0	0.23	1.9	2.14	2.81
As	0.82	0.23	0	1.67	1.91	2.58
President	1.49	0.9	0.67	2.34	2.58	1.91
Of	2.64	1.04	0.81	2.48	2.72	2.05
France	4.31	2.71	2.48	0.81	1.05	1.72

Table 2: Cost matrix of weighted edit distance

4.2. SUMMARIZATION

In the summarization based on the similar sentence detection, for a set of documents in a same topic, we first make clusters consisting of the similar word sequences included in the documents, then Generate summary by picking up a representative sentence from each cluster and put them in a specific order.

In this procedure, the clustering is the dominant part of computation. We developed an efficient clustering method specific to similar word sequence clustering.

Clustering

Step 1: Sub word sequence extraction

For each topic consisting of news articles belonging to the topic, we first extract all word sequences included in a sentence of the articles (See Fig. 4). Suppose the topic consists of news articles $\{D_1, D_2, \cdots, D_n\}$, each article D_i contains sentences $\{W_{i1}, W_{i2}, \cdots, W_{in_i}\}$ and each sentence W_{ij} consists of word sequences $w_{ij1}w_{ij2}\cdots w_{ijl_{ij}}$. Then, the set of word sequences is

$$S \equiv \bigcup_i \bigcup_{W_{ij} \in D_i} \bigcup_{k,l(1 \le k < l \le l_{ij})} \{w_{ijk} \cdots w_{ijl}\}. \qquad (6)$$

We decompose S into clusters.

Step 2: Indexing

To make clustering efficient, we construct an index for each word sequence W in S. The index is the permutation of the words in W according to the inverse document frequency defined by (1), i.e.,

$$idx(W) \equiv w'_1 w'_2 \cdots w'_l$$

where $w'_i \in W$ and $idf(w'_i) \geq idf(w'_j)$ holds for $1 \leq i < j \leq l$. We sort the set S of word sequences by alphabetical order of the index (see Fig. 5). Since the costs of edit operations are defined based on the inverse document frequencies (eqs. (3) to (5)), word sequences are dissimilar if they are apart from each other in the resultant sorted list.

Step 3: Redundant word sequence removal

By the definition of edit distance, if word sequences W_1 and W_2 have similar prefixes, any prefix of W_1 and W_2 also have similar prefixes. Therefore, we remove the word sequences in S that are a prefix of another word sequence. For each word sequence W in the sorted list obtained in step 2, if it is a prefix of adjacent word sequence in the list, we remove W. We denote the resultant list as $I \equiv \{W_1, W_2, \cdots\}$.

Step 4: Clustering

We make clusters using the index I. Let us first define the similarity between a word sequence W and a cluster C of word sequences. If W and a word sequence in C have similar prefixes, we can consider W is similar to the cluster C. According to this observation, we define the similarity between a word sequence W and a cluster C as the maximum length of the longest similar prefixes, i.e.,

$$sim(W, C) \equiv \max_{Z \in C} len(W_*, Z_*). \qquad (7)$$

We make clusters by assigning each word sequence to a cluster according to this similarity. For a set T of clusters, we assign a word sequence W to the most similar cluster in T if the similarity is larger than a threshold e. Otherwise, W constitutes a new cluster. In this procedure, we need to calculate the similarity (7) for all clusters in T. If the number of clusters is large, this process results in high computational cost.

As we mentioned above, the word sequences are dissimilar if they are apart from each other in the index obtained in step 3. Therefore, it is sufficient to consider the clusters that consist of the word sequences close to the objective sequence in the index. Let us define those candidate clusters. Let $d_i(W, Z)$ denote the distance of the word sequence W and Z in the index I, i.e., $d_i(W, Z) \equiv |i - j|$ if W and Z are i-th and j-th word sequences in I, respectively. For a cluster C and a word sequence W, we define the distance between C and W as the minimum distance of W and Z in C

$$d_i(W, C) \equiv \min_{Z \in C} d_i(W, Z).$$

Let λ be a parameter defining the sufficient number of clusters to be checked in the cluster assignment. Then, for a set T of clusters and a word sequence W, we refer to λ clusters that is most closest to W as candidate clusters.

Now, let us introduce the clustering algorithm. In the following algorithm, *Cand* keeps candidate clusters. As described in the algorithm, we scan the index I once and for each word sequence W in the index, we calculate the similarity between W and at most λ clusters. Therefore, we need O($\lambda|I|$) similarity calculations of eq. (7). Clustering usually requires quadratic calculations with respect to the number of objects. On the other hand, the proposed algorithm is linear to the object size because the parameter λ is much less than the index size. Therefore, it reduces the computational cost significantly. We show the efficiency of the proposed method by experiments.

Clustering Algorithm

Input: the index I defined in step 2.
Output: clusters T of word sequences
$T \leftarrow \{\}, Cand \leftarrow \{\}$
for i= 1 to $|I|$
  $W \leftarrow$ i-th word sequence in I
  find the cluster $C \in Cand$ most similar to W
    in terms of the similarity (7)
  if $sim(W, C) > e$, add W to C
    otherwise $T \leftarrow T \cup \{W\}, Cand \leftarrow Cand \cup \{W\}$
  if $|Cand| > \lambda$
    remove the farthest cluster from W
return T

Summary Generation

We generate summary from the set of clusters obtained by the clustering. For summary generation, we choose one representative sentence per cluster. Let $s(W)$ denote the sentence from which the word sequence W is extracted. Then, the representative sentence of a cluster C is defined as the longest sentence.

It is usually required to generate a summary whose length is shorter than the specified length. We choose sentences from larger clusters and concatenate the sentence until the summary length exceeds the specified length (See Fig. 8). The procedure of the summary generation procedure is summarized as follows:

Input: a set T of word sequence clusters and length τ
Output: summary
Sort T in the descending order of the cluster size.
Set *Summary* to null string
repeat while $|Summary| < \tau$
  Obtain the representative sentence S
    for the i-th cluster in the sorted list

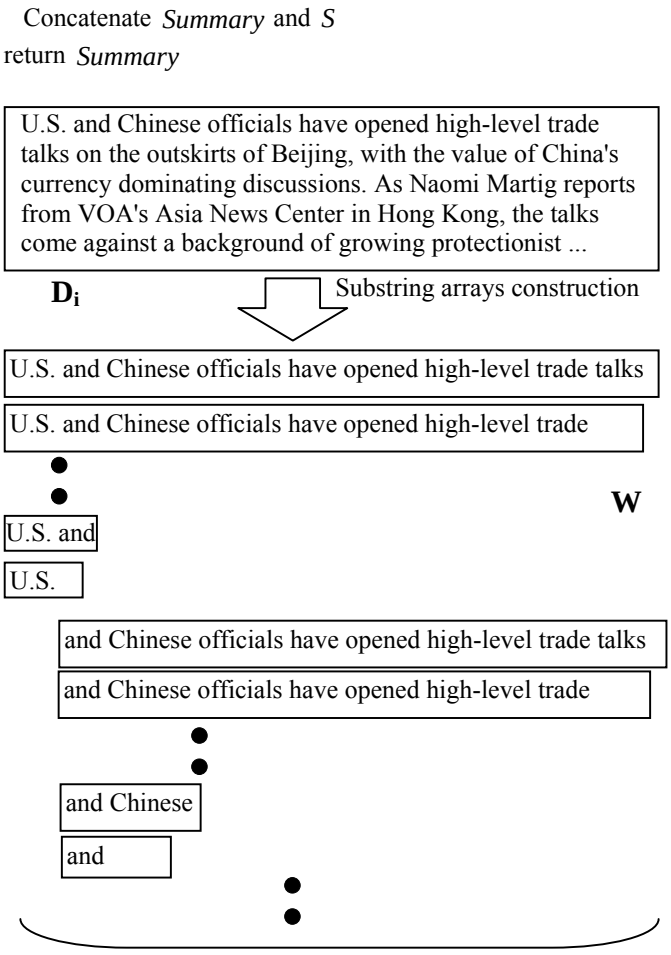

Figure 4: Enumeration of all possible substrings

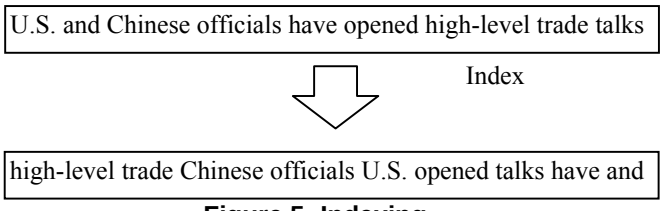

Figure 5: Indexing

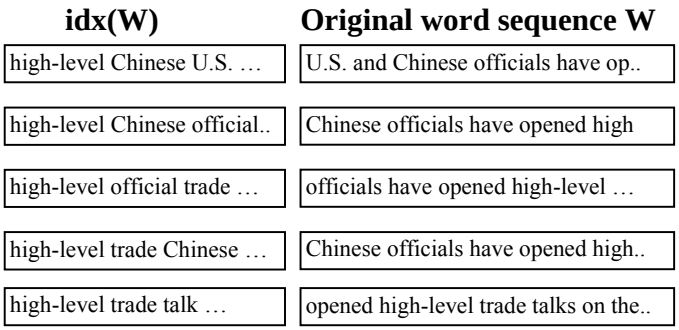

Figure 6: Sorted list of indices

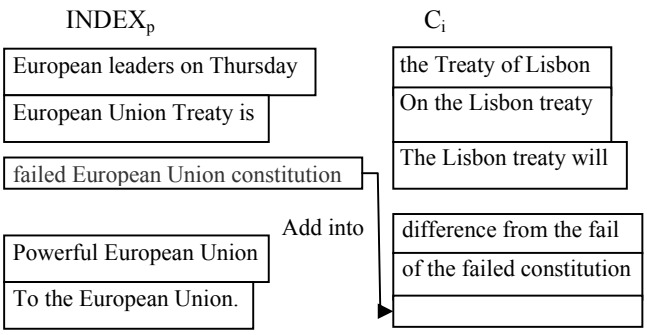

Figure 7: Summary appending

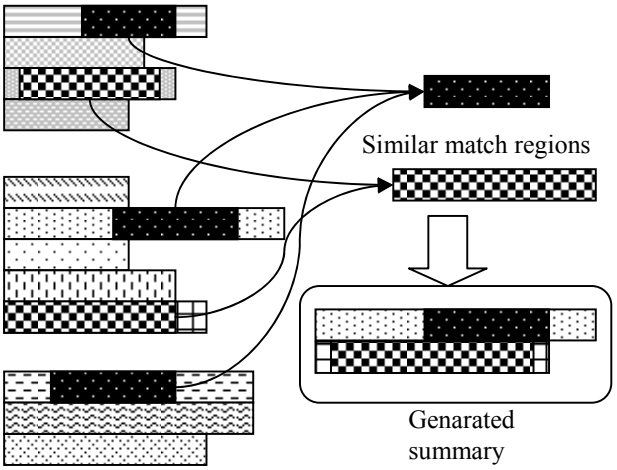

Figure 8: Summary appending

5. EXPERIMENTAL RESULTS

5.1. Data Set

We evaluated proposed method using the NTCIR4-TSC3 [17] corpus for the summarization. The corpus consists of articles from the Mainichi and Yomiuri newspapers (in Japanese) published between 1998 and 1999. The Corpus consists of 30 clusters of news articles. Each cluster corresponds to a topic. Average number of articles in a cluster is about 10. Hereafter we refer to the cluster as a topic. In the corpus, a set $\{m_1, m_2, \cdots, m_n\}$ of important sentences is manually assigned to each topic for both short and long summaries. The ideal summary should contain the information described by important sentences. Information may be described in a different way. Therefore, for each important sentence m_i, the corpus also provides a set $A_i \equiv \{A_{i,1}, A_{i,2}, \cdots, A_{i,l}\}$ of sentences in the news articles where an equivalent sentences $A_{i,j}$ is a set of sentences describing the information equivalent to m_i. In addition, the corpus provides manually written short and long summaries.

5.2. Quality of Summary

In NTCIR4-TSC3, two kinds of evaluation metrics were prepared. One is subjective, i.e., the scores were given by humans who read the generated summaries. The other is objective metrics called precision and coverage that can be calculated automatically by comparing the summary generated by system with the important sentences prepared manually. In this experiment we used these two objective metrics to compare the quality of summaries.

Precision is the ratio of how many sentences in the summary generated by system are included in the manually prepared important sentences [17]. Let h be the minimum number of sentences required for making a summary containing all information, and m be the number of important sentences included in the summary generated by system. Then, the precision is defined as

$$\text{Precision} = \frac{m}{h} . \quad (7)$$

Coverage is "an evaluation metric for measuring how close the system output is to the abstract taking into account the redundancy" [17] of the summary generated by system. For each important sentence m_i, let us consider the ratio that how many corresponding sentences are included in a generated summary E. Formally, for a set A_i of equivalent sentences for m_i, it is defined as

$$e(i) = \max_{1 \le j \le l} \left(\frac{|A_{i,j} \cap E|}{|A_{i,j}|} \right) . \quad (8)$$

Note that e(i) is 1 when any $A_{i,j}$ is completely included in the summary E. Using this ratio, the coverage is defined as

$$\text{Coverage} = \frac{\sum_{i=1}^{n} e(i)}{n} . \quad (9)$$

In this experiment we compared the proposed method with other summarization system. Important information is often described in the first part of document in news article. The baseline summarization method is a summarizer that takes the first part of the news article as important sentences. This method is referred to as "LEAD" in this paper. LEAD is the method to make extracts from head of article. It picks a sentence one by one from the head of article in order of time for Multi-document summarization (TSC3).

The second groups of compared systems are those that participated in TSC3. There were 10 systems that are referred to as "SOKEN a", "SOKEN b", "CRLNYU a", "CRLNYU b", "smlab" , "MOGS", "forest", "DBLAB", "UEC" and "UYDI". Many of them used linguistic perspective (stemming, stop word removing or event modeling). Since the proposed method is based on the similar sentence detection, we also compared the proposed method with the ClustalW [19], which is often used to make alignments of strings and genome sequences.

Table 3 shows the result of summarization. The results of the second group, i.e., systems participating in the TSC3, are copied from the article [17]. First the proposed method is slightly superior to the baseline method LEAD. Compared with the systems participating in the TSC3, however, the proposed method could not achieve better performance. ClustalW has similar feature. Both the proposed method and ClustalW are categorized into the similar sentence detection method. From this result, the summarization method based on the similar sentence detection has disadvantages in finding important sentences. Note that the proposed method outperforms ClustalW because it uses weighted edit distance based on inverse document frequencies.

	Short		Long	
	Coverage	Precision	Coverage	Precision
SOUKEN a	0.315	0.494	0.355	0.554
SOUKEN b	**0.372**	**0.591**	0.363	0.587
CRLNYU a	0.222	0.314	0.313	0.432
CRLNYU b	0.293	0.378	0.295	0.416
smlab	0.328	0.496	0.327	0.535
MOGS	0.283	0.406	0.341	0.528
forest	0.329	0.567	**0.391**	**0.68**
DBLAB	0.308	0.505	0.339	0.585
UEC	0.181	0.275	0.218	0.421
UYDI	0.251	0.476	0.247	0.547
LEAD	0.212	0.426	0.259	0.539
ClustalW	0.209	0.341	0.247	0.536
Proposed method	**0.252**	**0.437**	**0.277**	**0.522**

Table 3: Coverage and Precision of Summarizers

Both coverage and precision are defined to measure how much the generated summary contains the information required for the summary. However, they are not suitable for measuring redundancy of the generated summary. Therefore, we also used an evaluation metric redundancy on important sentences (RIS) in addition to precision and coverage [20]. RIS is defined as follows. For each important sentence m_i, let L_i be union of equivalent sentences for m_i, i.e.,

$$L_i = \bigcup_{j=1}^{l} A_{i,j} . \quad (10)$$

For a summary E generated by a system, let $S_i^{\min}$ be the minimum subset of E that satisfies $e_i(E) = e_i(S_i^{\min})$. Then, RIS for the summary E is defined as

$$\text{RIS}(E) = \frac{\sum_{i=1}^{l} (|E \cap Li| - |S_i^{\min}|)}{l} . \quad (11)$$

Note that lower RIS means the better summary with respect to the redundancy. Table 4 shows the RIS of summarizers. Since there is no report on the evaluation of systems participating in TSC3 with RIS, table 4 only contains the LEAD, ClustalW and the proposed method.

Table 4 shows that both ClustalW and the proposed methods are much better than LEAD. This indicates that

the similar sentence detection methods are effective in reducing the redundancy on important sentences.

	RIS short	RIS long
LEAD	0.236113445	0.280692756
ClustalW	0.086785714	0.175450614
Proposed method	0.097370837	0.111142954

Table 4: Redundancy on important sentences

5.3. Processing Efficiency

Natural language processing usually requires complicated computations and it requires high computation cost. For example, computational complexity of parsing sentence is $O(n^3)$ whereas the measurement of the sentence similarity based on the edit distance can be done in $O(n^2)$ time complexity. Therefore, we consider the similar sentence detection approach may be more efficient than the methods using linguistic knowledge.

We cannot compare the processing time of the systems described in the previous section because they are not available. Therefore, we compare the proposed method and ClustalW in this section.

We measured the processing time for making the summary from a set of news articles belonging to the same topic. Figure 9 shows the processing time with respect to the number of letters included in the news articles in a cluster. As shown in the figure, the proposed method is much faster than ClustalW. ClustalW requires $O(|S|^2)$ calculations of sentence similarity, whereas the proposed method requires $O(\lambda|I|)$ calculations that is linear to the number of sentences. This is the main reason why the proposed method is faster than ClustalW.

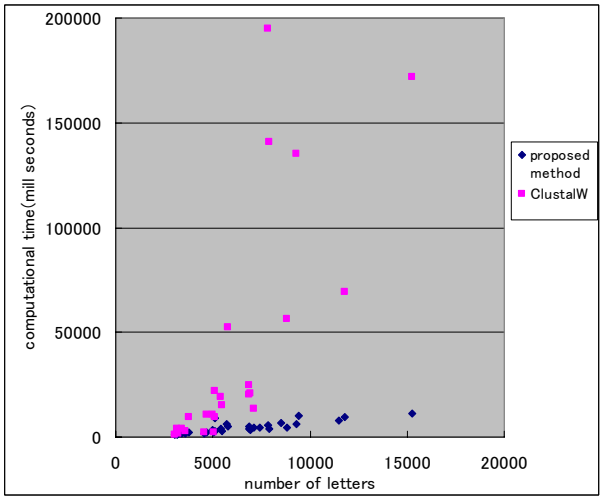

Figure 9: Computational time

6. CONCLUSION AND FUTURE WORKS

This paper proposes a news article clustering and summarization system. The proposed system achieves efficient news topic summarization. The proposed method generates good summaries in terms of the redundancy on important sentence; however, it requires improvement in detecting important sentences.

We plan to improve the ability to detect important sentence by incorporating the term frequencies (TF) as well as document frequencies. TF is useful feature to handle content of documents and it can be measured efficiently. Also, we would like to compare computational time and RIS of proposed one to other multi-document summarization methods in order to show reasonable efficiency. Additionally, the proposed method can be used to find arbitrarily similar strings, and therefore, it has many applications. We want to improve our method in order to find the similarity regions that appear in many documents at the same time.

We plan to adjust the method to cover multi-language news articles and also plan to adjust any corrupt languages, such as weblog. We plan to discuss these issues in the future.

References

[1] Google News http://news.google.com/

[2] Columbia Newsblaster http://newsblaster.cs.columbia.edu/

[3] U. Manber and G. Myers, Suffix ars, Vol 6, pp. 132–137.

[4] S Kurtz. Approximate String Searching under Weighted Edit Distance. In Proc. of Third South American Workshop on String Processing, 1996 pp. 156-170

[5] TDT http://www.nist.gov/speech/tests/tdt/

[6] Jaime Carbonell and Jade Goldstein. The use of MMR, diversity-based reranking for reordering documents and producing summaries. In Proceedings of ACM-SIGIR'98, Melbourne, Australia, August 1998.

[7] A. Apostolico, 1985. The myriad virtues of subword trees. In Combinatorial Algorithms on Words. Springer-Verlag, Barlin, pp. 85–96.

[8] E. Ukkonen, Finding approximate patterns in strings. J. Algor. Journal of Algorithmniversity of Waterloo.

[9] S. Wu and Manber. Fast text searching allowing errors. Communications of the ACM 35, 10, pp. 83–91.

[10] R. Baeza-Yates, 1989. Efficient Text Searching. Ph.D. thesis, Dept. of Computer Science, Urays: a new method for on-line string searches, in 'SODA' 90, pp. 319-327

[11] Dragomir R. Radev, S. Blair-Goldensohn, Z. Zhang, R. S. Raghavan. NewsInEssence: A system for domain-independent, real-time news clustering and multi-document summarization. In Human Language Technology Conference (Demo Session), San Diego.

[12] Dragomir R. Radev, Hongyan Jing, and Malgorzata Budzikowska, 2000. Centroid-based summarization of multiple documents: sentence extraction, utility-based evaluation, and user studies. In Proc. of the ANLP/NAACL Workshop on Automatic Summarization.

[13] J. Goldstein, V. Mittal, J. Carbonell, and M. Kantrowitz. Multi-document summarization by sentence extraction. In Proceedings of the ANLP'2000 Workshop on Automatic Summarization, 40–48. New Brunswick, New Jersey: Association for Computational Linguistics.

[14] R. Barzilay, K. McKeown, and M. Elhadad. Information fusion in the context of multi-document summarization. In Pro. of ACL-99.

[15] K. R. McKeown, R. Barzilay, D. Evans, V. Hatzivassiloglou, J. L. Klavans, C. Sable, B. Schiffman, and S. Sigelman. Tracking and summarizing news on a daily basis with Columbia's newsblaster. In Proceedings of the Human Language Technology Conference, 2002.

[16] V. Hatzivassiloglou, J. L. Klavans, M. L. Holcombe, R. Barzilay, M.-Y. Kan, and K. R. McKeown. SIMFINDER:A flexible clustering tool for summarization. In NAACL Workshop on Automatic Summarization, pp. 41-49. ACL, 2001.

[17] T. Hirao, M. Okumura, T. Fukushima, and H. Nanba. Text Summarization Challenge 3 –Text Summarization Evaluation at NTCIR Workshop4- Working Notes of the Fourth NTCIR Workshop Meeting, 2004.

[18] T. Takeda and T. Takasu. UpdateNews: a news clustering and summarization system using efficient text processing. International Conference on Digital Libraries Proceedings of the 2007 conference on Digital libraries.

[19] Julie D. Thompson, Desmond G. Higgins, and Toby J. Gibson. CLUSTAL W: improving the sensitivity of progressive multiple sequence alignment through sequence weighting, position-specific gap penalties and weight matrix choice. Nucleic Acids Research, 1994, Vol. 22, No. 22 pp. 4673-4680

[20] T. Hirao, M. Okumura, T. Fukushima, H. Nanba, C. Nobata, and H. Isozaki. Corpus and Evaluation Measures for Extractive Multiple Document Summarization. IPSJ TOD 35 Vol.48 No.SIG14. pp. 60-67.