

Corpus-Based Chinese-Korean Abstracting Translation System

Jun-Jie Li and Key-Sun Choi

CSLab, Center for AI Research, Korea Advanced Institute of Science and Technology
Taejeon, Republic of Korea

E-mail: {jkleee,kschoi}@world.kaist.ac.kr

Abstract

A Corpus-Based Chinese-Korean Abstracting Translation System is designed and implemented. Firstly, a text indexing method called Natural Hierarchical Network(NHN) is introduced, and then a Corpus-Based Word Segmentation algorithm is developed with the segmentation correctness of 98% for open test. Based on a words weighting function and a sentence importance weighting function which can dynamically calculate the importance of words and sentences by using the word frequency both in corpus and context, word length, sentence length and so on, an abstracting system is implemented to produce abstracts of texts in deferent languages and domains by any abstracting rate. Experiments show that generally abstracts produced by 10% to 20% abstracting rates can cover 90% of the important sentences of the input texts. Finally, combines with an Example-Based Chinese-Korean Machine Translation System, the generated abstracts are translated into target language with the correctness of translation of more than 70% by the important words oriented machine translation strategy.

1 Introduction

Automatic Abstracting System is a very attractive, historical and difficult topic of Natural Language Processing. Its aim is to identify and select the central content or user inquired content from the given original texts to form the summarized output with the sentences identical to the original input text or new generated. There are three kinds of methods on developing Abstracting System: the first one is based on the surface-clues of the current context such as the word frequency [Luhn, 1958], sentence position, word clue or indication, title sentence[Watanabe,1996], word association or rhetorical relations[Ono *et al.*,1994] and linear heuristic sentence weighting function [Zechner, 1996]. Its advantages are simple and domain unconstrained, its shortcoming is inaccuracy in sentence abstracting due to the uncertain value of word frequency for key words, varied distribution of important sentences and heuristic function itself. The second one is based on the knowledge-based natural language processing tech-

niques, such as Script-based summarization system for given texts with multilingual output[Tait, 1985], CD-based domain constrained abstracting system with incomplete syntactic and semantic analysis [Dejong, 1979], rule-based summarization system with forward and backward scanning schema [Danilo,1982]etc.. Its advantages are more accurate and in depth language analysis and generation. Its shortcomings are domain constrained and difficulty in knowledge base maintenance. The third one is the corpus based methods [Li and Wang, 1995; Li, 1995]. The corpus based sentence segmentation, non-linear sentence weighting function, collocation computation based word and sentence importance analysis and efficient raw corpus and text indexing method, give this method a prospective future.

Example Based Machine Translation system(EBMT) is essentially translation-by-analogy: given a source-language passage S and a collection of aligned source/target text(or sentence) pairs, find the "best" match for S in the source -language half of the text collection, and accept the target-language half of that match as the translation[Brown, 1996].

In this paper, we firstly introduce the methodology of Automatic Abstracting System and Example-based MT system, and then in section 2, a corpus indexing method called Natural Hierarchical Network(NHN) is illustrated. In section 3, the corpus based word segmentation algorithm is introduced. In section 4, the word weighting function and sentence weighting function as well as abstract generation algorithms are introduced in detail. In section 5, an EBMT system called EBMT/CK is illustrated which is to translate Chinese abstracts into Korean. Finally the experimental results and conclusion are given in the section 6 and 7.

2 Natural Hierarchical Network

Natural Hierarchical Network(NHN), shown as Figure 1, is a hierarchical weighted direct graph. Let T_i , $i=1,2,\dots,n$, denotes the collection of members of level i . The elements in T_i is denoted as t_{ij} . Now, we define a set of mappings $f_i : T_{i+1} \times N \rightarrow T_i$, $i=1,\dots,n-1$, where N is the natural number. For $\forall t_{i+1j} \in T_{i+1}$, $t_{ik} \in T_i$, $f_i(t_{i+1j}, w_{ip}) = t_{ik}$ represents that t_{ik} is the w_{ip} -th component of t_{i+1j} , and $\exists t_{i1}, t_{i2}, \dots, t_{im} \in T_i$, $t_{i+1j} = t_{i1}, t_{i2}, \dots, t_{i w_{ip}-1} t_{i w_{ip}} t_{i w_{ip}+1} \dots t_{im}$.

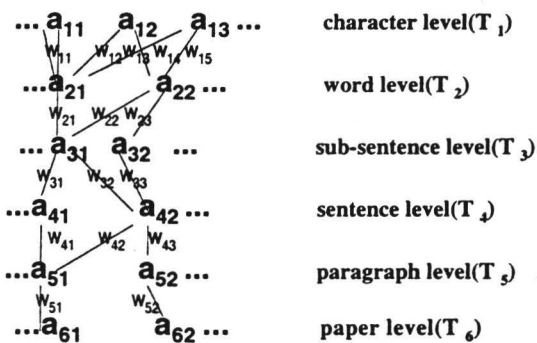


Fig.1 Description of Natural Hierarchical Network

Since $f_i(t_{i+1j}, w_i) = t_{ik}$ corresponds to an weighted edge $E((t_{ik}, t_{i+1j}), w_{ip})$ in NHN. In general, Let $(t_{mk}, w_m, t_{m+1j}, w_{m+1}, \dots, w_{n-1}, t_{nl})$ be a path from t_{mk} to t_{nl} , then it can be represented by a vector $(l, w_{n-1}, \dots, w_{m+1}, w_m, t_{mk})$, where $t_{mk} \in T_m$, $t_{nl} \in T_n$, w_i is the weight of edge (t_{ik}, t_{i+1j}) , $i \in [m, n-1]$.

The meaning of NHN is that every language unit(character, word, sentence, paragraph) have a vector to corresponds to its every occurrence in texts, and in turn, the texts(or raw corpus) can be indexed and represented by all the occurrences of elements in a certain level m (say character level) represented by the vectors as above. Certainly, we can omit some levels to make the vector shorter, however that will lose some useful information of text structure and language usage.

In practice, according to the sentence and sub-sentence ending symbols(i.e., punctuation such as “ . , : ? ! ”) and the format of text(such as writing rules or custom of paragraph, chapter, title and subtitle), the input text can be automatically converted into a series of vectors as $(pp, pa, sn, ss, wd, ct, c_i)$, where pp, pa, sn, ss, wd and ct are respectively represent the sequential numbers of paper, paragraph, sentence, sub-sentence, word and character that character c_i appears in.

For Chinese(similar to Japanese and Korean), on indexing the non-segmented texts, the “wd” element of the above vector will always be 1 if this element is considered useful for the later application, otherwise we may just omit the “wd” element.

Algorithm createNHN(t)

```
{
/*given a input text t, transform it into NHN representation by analyzing text structure and yield NHN for each character and sentence and store them into data base.
*/
```

Step 0. Initialization. Assign the paper number pp an identical value for the current input text. Let paragraph number $pa=1$, sentence number $sn=1$, sub-sentence number $ss=1$, word number $wd=1$, character number $ct=1$. Let sentence buffer be empty. Read the input text into memory.

Step 1. let Ch be the current processing character .

Step 2. If the character is a sentence ending symbol such as “ . ? ! ”, store the current sentences and its NHN vector (pp, pa, sn, ss) into the sub-sentential index-

ing data file. then if there is no line return symbol follows it, it means this is the end of the sentence, and let $sn=sn+1, ss=1, wd=1, ct=1$, to prepare for the next sentence process ; Otherwise, it implies that this is the end of the paragraph, then let $pa=pa+1, sn=1, ss=1, wd=1, ct=1$.

Step 3. If the character is a subsentence ending symbol such as “ , ; : ”, store the current sub-sentences in the sentence buffer and its $NHN(pp, pa, sn, ss)$ into the sub-sentential indexing data file. Then let $ss=ss+1; wd=1; ct=1$.

Step 4. If the character is a blank character, which means that the current word ends, then let $wd=wd+1; ct=1$.

Step 5. If the character is not the above punctuations, and it is a normal language character, then store the character and its $NHN(pp, pa, sn, ss, wd, ct)$ into character indexing data base.

Step 6. If there are still characters unprocessed, go to Step 1.

3 Corpus-Based Word Segmentation

Algorithm Segment(s)

```
{
/* given a string s(initially is the whole sentence), segment it into some words.
*/
```

Step 0. Initialization. Let max represents the current maximum weight value, initially $max=0$;

Step 1. Computing weights of all the substrings in s .

Step 2. Pick up the string with the greatest weight, say s' , which is the current abstracted word and store it. If s' equals to s , then exit, otherwise go to step 3.

Step 3. Segment($s-s'$), $s-s'$ is the left strings.

Algorithm Weighting(s)

```
{
/*Given a string  $s = c_1 c_2 \dots c_n$ , in order to compute weights of all strings in  $s$ , it is necessary to build a collocation matrix  $A$ , where  $A(i, j)$  represents the frequency of string from character  $i$  to  $j$ , then the weight of that string can be calculated by the weighting function  $W(c_i c_{i+1} \dots c_j) = F(c_i c_{i+1} \dots c_j) * (j-i+1)^c$ , where  $F(c_i c_{i+1} \dots c_j)$  is the frequency of string  $c_i c_{i+1} \dots c_j$ , and its length is  $(j-i+1)$ ,  $c$  is a constant power of length,  $c > 1$ . In practice when  $c$  equals 3, the words abstracted are more probable to be correct;
*/
```

Step 0. Initialization. Search the data base to find out NHN set T_i of each $c_i, i = 1, \dots, n$.

Step 1. For $(j=1; j \leq n-1; j++)$

Step 2. For $(i=1; i \leq j-1; i++)$

Step 3. $T_{ij} = T_{i,j-1} \wedge T_j$; // to compute collocatoin of column j in matrix A

Step 4. $A(i, j-1) = W(c_i c_{i+1} \dots c_j) = |T_{i,j-1}| * (j-i)^c$; //to weight the string $c_i c_{i+1} \dots c_{j-1}$.

where, T_{ij} is the NHN set of $c_i c_{i+1} \dots c_j$, $T_{ij} = T_{i,j-1} \wedge T_j = (((T_i \wedge T_{i+1}) \wedge T_{i+2}) \wedge \dots) \wedge T_j$, “ $\wedge$ ” means collocation computation. For example, let $(pp_1, pa_1, sn_1, ss_1, wd_1, ct_1) \in T_i$,

$(pp_2, pa_2, sn_2, ss_2, wd_2, ct_2) \in T_{i+1}, \text{if}((pp_1=pp_2) \& \& (pa_1=pa_2) \& \& (sn_1=sn_2) \& \& (ss_1=ss_2) \& \& (wd_1=wd_2) \& \& (ct_1+1=ct_2))$, then it means that c_i and c_{i+1} collocate once with c_i appearing to the left side of c_{i+1} . let $(pp_2, pa_2, sn_2, ss_2, wd_2, ct_2) \in T_i \wedge T_{i+1} = T_{i,i+1}$.

In practice, multiple segmentation technique is utilized with first scanning of segmentation being based on the computation of string frequency in context to find the unknown words and solving ambiguous segmentation and second scanning of segmentation being based on the frequency in corpus to segment common words.

4 Word and Sentence Weighting Function and Abstract Generation

4.1 Word Weighting Function

The words in context have different importance and contribution for the theme. In general, functional words(such as proposition, pronoun, conjunction) are less important than the content words(such as noun and verb). The characteristics of functional words are high frequency of usage and shorter length, while the content words are just the opposite, with a lower frequency of usage and longer length. In addition, the important words(often called key words), often have a certain higher frequency of appearance in context and lower frequency of appearance in the corpus, while the functional words have higher frequency of occurrence in both the context and the corpus. Based on these characteristics, the word weighting function is designed as follows:

$T(w) = (F_1(w)/F_2(w)) * (L(w))^c$,
where $F_1(w)$ is the frequency of w in context, $F_2(w)$ is the frequency of w in corpus, $L(w)$ is the length of w , c is a constant power of length, in practice $c=3$.

The purpose of using $L(w)$ is to give the longer word a higher weight than the shorter words because longer words with a relatively same high frequency as shorter words in the context often indicate more important than shorter words. By this weighting function, the important words will be gained higher weight.

4.2 Sentence Weighting Function

The important sentences generally illustrate themes or topics of contexts in a condensed and conclusive way, such as title and subtitle sentences, topic sentences and other conclusive sentences. The characteristics of important sentences are generally to contain more important words (or key words) and have a shorter sentence length and few number of sub-sentences. Therefore, the sentence weighting function is designed as follows:

$P(s) = (T(w_1) + T(w_2) + \dots + T(w_n)) / (L(s) * N(s))$,
where s is a sentence and w_i is a word of s , $T(w_i)$, $i=1, \dots, n$, is the weight of w_i , $L(s)$ is the length of sentence s , $N(s)$ is the number of sub-sentences in s .

According to this function the shorter sentences with more important words will be given higher weight, so title and subtitle sentences, topic sentences and most of the conclusive sentences will have more chance to obtain higher weights than the other unimportant sentences.

There are also other interesting factors such as digital numbers, sentence locations and word clues. this kind of factors is often user-oriented and text style related, and if used properly, it will make good effect.

4.3 Abstract Generation

The abstracts(or summaries) are generated by selecting the important sentences with higher sentence weight from the input text, and keeping their original sequential orders in the text.

Algorithm Abstract(t, r)

```
{
/* t is a input source text, r is the summarizing rate,
then generate the abstract A (t) of t */
Step 1. Initialization. Let the length of abstract
L(A(t))=0; Let the sentences of t be reordered in a
queue,  $s_1, s_2, \dots, s_n$ , according to their weight with the
former sentences having higher weight than the later
sentences in the queue. Let counter  $i=1$ .
Step 2. Select a sentence  $s_i$ , if  $L(A(t)) + L(s_i) \leq$ 
 $L(t)*r$ , then put into A(t) and  $i=i+1$ ; otherwise exit.
Step 3. Go to step 2.
}
```

5 Example-Based Chinese-Korean Machine Translation System

The purpose of the EBMT/CK system is to translate the abstracts produced by the above Abstracting System into target language. In practice, we use Chinese as the source language and Korean as the target language. The sentences and words in the abstracts have been segmented and weighted by the previous modules, then the main tasks for EBMT/CK is to select pattern sentences from bilingual sentential-alignment example corpus and conduct word-alignment as well as generate target translations.

EBMT/CK uses essentially no knowledge about its source or target language. Its three knowledge sources are: a sententially-aligned bilingual example corpus; a bilingual dictionary; a Chinese-Korean Character Transforming Table.

5.1 Pattern Sentences and Optimal Cover Set Finding

Algorithm OptimalCoverSet(S)

```
{
/*Given a segmented source sentence,  $S = w_1 w_2 \dots w_m$ ,  $w_i$ 
is the segmented words or phrases. Let  $W(w_i)$  is the
weight of  $w_i$ , computed by the same word weighting
function as in section 3,  $W(w_i) = F(w_i) * L(w_i)^c$ , however,
the difference is that the word frequency is computed
in Bilingual corpus. Let  $T(w_i)$  be NHN vectors col-
lection of  $w_i$ . Then, we compute weight of each sen-
tence that corresponds to a NHN vector in  $T(w_i)$ , and
select a set of example sentences in  $T(w_i)$  to cover the
sentence  $S$ .*/
```

Step 1. Compute weight of each NHN vector in $T(w_i)$ which corresponds to a example sentence in Bi-lingual corpus, $i=1, \dots, m$; if $w_{i1}, w_{i2}, \dots$ and w_{ik} occur in the example sentence(say e_1), represented by a vector

(pp,pa,sn,ss) , then $P(e_1) = (W(w_{i1}) + \dots + W(w_{ik}))/L(e_1)$, and let $C(e_1) = \{w_{i1}, w_{i2}, \dots, w_{ik}\}$.

Step 2. Select the sentence with the greatest weight, let it be e_1 .

Step 3. If $\{w_1, w_2, \dots, w_m\} - C(e_1) \neq \Phi$, then continue to select the sentence greatest weight from the left sentences, until we get a set of sentences $\{e_1, e_2, \dots, e_k\}$, and $C(e_1) \cup C(e_2) \cup \dots \cup C(e_k) \supseteq \{w_1, \dots, w_m\}$.

Step 4. Output $\{e_1, \dots, e_k\}$ as the optimal cover set of sentence S , and e_1 is the pattern example sentence because of its greatest weight.

}

5.2 Word-Alignment

Word alignment uses an approximate match approach, because 60% of the Korean words (most of them are noun and verb with word length greater than or equal to 2) are originated from Chinese with the same or similar meanings and lengths but written in the form of Korean pronunciation (Hangul) character.

Therefore, we firstly translate every Chinese character in pattern sentence into corresponding Korean pronunciation character by utilizing a Chinese-Korean Character Transformation Table in which every Chinese character has a corresponding Korean pronunciation representation such as Chinese character “李” in Chinese GB code corresponds to Korean pronunciation representation “o|” in Korean KSC code. Then approximate word matching between the transformed words and the words in Korean example sentence of the pattern sentence pair is conducted by using the following word similarity function:

$S(w_1, w_2) = (\text{the number of characters in both words}) * 2 / (L(w_1) + L(w_2))$.

In this way, more than 60% of the words can be aligned, most of them are the Korean words originated from Chinese. Some unknown words such as the name of people, organization and so on, can also be aligned; Besides, one-to-many, many-to-one and one-to-one associations can also be solved by using the following word alignment algorithm.

Algorithm WordAlignment(p_1, p_2)

{/* $p_1 = w_{11}w_{12} \dots w_{1m}$ and $p_2 = w_{21}w_{22} \dots w_{2n}$ is an example sentence pair, p_1 is Chinese (source language) half, while p_2 is Korean (target language) half of the pair, w_{1i} and w_{2j} are words or phrases, $i \in [1, m]$, $j \in [1, n]$. */

Step 1. Transform $p_1 = w_{11}w_{12} \dots w_{1m}$ into $p'_1 = w'_{11}w'_{12} \dots w'_{1m}$ by looking up Chinese Character Transformation Table.

Step 2. For every $i \in [1, m]$, $j \in [1, n]$, Compute $S(w'_{1i}, w_{2j})$

Step 3. For a given $i \in [1, m]$, if exist $k_1, \dots, k_h$, $h \geq 1$, for every $j \in [1, n] - \{k_1, \dots, k_h\}$, $S(w'_{1i}, w_{2kh}) = \dots = S(w'_{1i}, w_{2kh}) > S(w'_{1i}, w_{2j}) > 0$, then w'_{1i} is aligned by $\{w_{2k1}, \dots, w_{2kh}\}$.

Step 4. Otherwise, if no such k exists (i.e. for every $j \in [1, n]$, $S(w'_{1i}, w_{2j}) = 0$), then look up w_{1i} in a bilingual dictionary of common words, let the translations of w_{1i} be $w_{1i}^{(1)}, \dots, w_{1i}^{(k)}$, $k \geq 1$, then for every $j \in [1, n]$, $h \in [1, k]$, compute $S(w_{1i}^{(h)}, w_{2j})$. If exist $\{j_1, \dots, j_p\} \in [1, n]$ and $\{h_1, \dots, h_p\} \in [1, k]$, $S(w_{1i}^{(h1)}, w_{2j1}) = \dots = S(w_{1i}^{(hp)}, w_{2jp}) > S(w_{1i}^{(h)}, w_{2j}) > 0$, where $h \in [1, k] - \{h_1, \dots, h_p\}$, $j \in [1, n] - \{j_1, \dots, j_p\}$, then w_{1i} is aligned by $\{w_{2j1}, \dots, w_{2jp}\}$.

Step 5. If w_{1i} is not listed in dictionary and for every $j \in [1, n]$, $S(w'_{1i}, w_{2j}) = 0$, then w_{1i} is aligned with $\{w_{2k1}, \dots, w_{2kh}\}$, where $w_{2kj} (j=1, \dots, h)$ has no association with w_{1i} , $i \in [1, m]$.

}

5.3 Target Sentence Generation

Based on the pattern sentence and word alignment, the target sentence is generated by replacing some of the words in pattern sentence so that the modified pattern sentence becomes identical to the input source sentence.

Algorithm TargetSentenceGeneration(s, p_1, p_2)

{/* s is the source sentence, p_1 and p_2 are pattern sentence pair, p_1 is the Chinese example sentence, and p_2 is the Korean translation of p_1 . To generate the Korean translation of s by modifying pattern sentence pair.*/

Step 1. Align s and p_1 . If there are k words (say, $w_1, w_2, \dots, w_k$) appear in both s and p_1 . Let $s = t_1w_1t_2w_2 \dots w_kt_{k+1}$, $p_1 = t'_1w_1t'_2w_2 \dots w_kt'_{k+1}$, then for $i \in [1, k]$, t_i is aligned with t'_i , where t_i and t'_i are words, phrases or blank.

Step 2. After calling WordAlignment(p_1, p_2), using t_i to replace $t'_i \in p_2$ which associate with t'_i .

Step 3. Translate $t_i (i \in [1, k])$ by first converting them into Korean by using the Transformation Table and then looking up dictionary.

}

There is one exception to the above procedure for retrieving and aligning chunks. If any of the chunks cover the entire input string and the entire source-language half of a corpus sentence pair, then all other chunks are discarded and target-language half of the pair is produced as the translation.

6 Experiment

6.1 Corpus Indexing

In practice, we give each character an entry and its every occurrence in texts is described by a vector as (paper number, paragraph number, sentence number, sub-sentence number, word number and character number), the indexing speed by algorithm CreateNHN(t) is about 2000 characters/sec on IBM PC-486. Apparently, storage space for the NHN data is about the six times of the original texts data space.

6.2 Word Segmentation

The corpus used for word segmentation is a collection of texts without restrictions on domain, style and length.

One experiment shows that the number of abstracted words by using algorithm Segment(s) is increased along with the size increment of corpus, for example when the size of corpus is 27000 characters, the average coverage rate of abstracting (which is calculated by (number of characters in abstracted words)/(number of characters in tested texts)) is about 86%, while most of the left 14% of unprocessed strings are already words, therefore further increment of the corpus size will not increase the coverage rate too much, and in general, the size of corpus around 50000 characters is enough

to obtain a realistic and steady coverage rate of abstracting.

Another experiment shows that the correctness of abstracting (which is calculated by $1 - (\text{the number of wrong abstracted words}) / (\text{the number of total abstracted words})$) is about 98% with the above coverage rate of abstracting and does not changed too much when the coverage rate of abstracting go beyond 80%.

An comparison of 30000 word dictionary based Backwards Maximum Matching(BMM) algorithm and the Non-dictionary Word Segmentation(NWS) algorithm, shows that the number of wrong segmented words of BMM is four times of that of NWS for open test.

The running speed is 2000 characters/min for the corpus with 40000 characters. The computational complexity of time for the algorithm is $O(m*n)$, m is the size of input text, n is the size of the corpus.

6.3 Abstract Generation

In order to evaluate the machine-made abstracts, we selects tens of articles from different domains with different styles and lengths.

One experiment shows that 90% of the abstracts of these articles with 5%-10% abstracting(or summarizing) rate can contain the title sentence and some subtitle sentences and topic sentences, although we do not give particular attentions and weights to those sentences in our word and sentence weighting function.

Another experiment shows that 90% of the important sentences neither title sentences nor topic sentences can also be abstracted by the abstracting rates less than 20%.

We have test our system on Chinese and Korean language texts, although most of the above experiments are conducted to Chinese texts, the Korean texts processing have the similar results on a small scale of test, since the methods we used are not language oriented.

6.4 Example-Based Abstracting Translation

We use a bilingual example corpus with 5000 Chinese-Korean sentence pairs, a Chinese-Korean dictionary of 5000 most frequently used words and a Chinese - Korean Character Transformation Table with each entry corresponding to a Chinese character.

An experiment shows that based on a small scale of text data, the correctness of word alignment is 80% and more than 70% of the important words can be correctly translated.

7 Conclusion

we have illustrated the detailed algorithms of Abstracting Translation System(ATS) and shown some of the experimental data. The key techniques include the NHN(Natural Hierarchical Network) raw corpus(and text) indexing method, the corpus- based word segmentation and word weighting function, dynamic sentence weighting function, and abstract generation algorithm, an Example-Based Chinese-Korean MT system with algorithms in pattern sentence finding, important word oriented word alignment and target sentence generation, etc.. The algorithms and ideas of our system are

also quite meaningful for Multilingual Information Retrieval and Corpus-based Natural Language Processing.

Acknowledgment

This project has been supported by China National High -Tech Development Plan "863" on Automatic Abstracting System and Center of AI Research of Korea Advanced Institute of Science and Technology on Chinese-Korean MT system.

References:

- [Watanabe, 1996] Hideo Watanabe. A Method for Abstracting Newspaper Articles by Using Surface Clues. COLING96:947-979, Copenhagen, Denmark, Aug. 5-9, 1996.
- [Zechner, 1996] Klaus Zechner. Fast Generation of Abstracts from General Domain Text Corpora By Extracting Relevant Sentences. COLING96:986-989, Copenhagen, Denmark, Aug. 5-9, 1996.
- [Luhn, 1958] Luhn, H.P.. The Automatic Creation of Literature Abstracts. IBM Journal of Research and Development, Vol.2, No.2:159-165, 1958.
- [Dejong, 1979] G. Dejong. Prediction and Substantiation : Two Processes That Comprise Understanding. Proceedings of IJCAI-79, Tokyo, Jan., 1979.
- [Tait, 1985] J.I. Tait. Generating Summaries Using a Script-Based Language Analyzer. Progress of Artificial Intelligence, 1985.
- [Danilo Fum. et al, 1982] . Danilo Fum. et al. Forward and Backward Reasoning in Automatic Abstracting. COLING82.
- [Ono et al., 1994] Kenji Ono, Kazuo Sumita and Seiji Miike. Abstract Generation Based on Rhetorical Structure Extraction. COLING94, Vol.1:344-348, Kyoto, Aug. 5-9, 1994.
- [Li and Wang, 1995] JunJie Li and KaiZhu Wang. Study and Implementation of Non-dictionary Chinese Segmentation. NLPRS'95, Seoul, Korea, Dec., 1995.
- [Li, 1995] Li Junjie. Research and Implementation of Unconstrained Chinese Automatic Abstracting System. Ph.D. dissertation, Harbin Institute of Technology, P.R.China. Aug., 1995.
- [Brown, 1996] Ralf D. Brown. Example-Based Machine Translation in the Pangloss System. COLING96:169-174, Copenhagen, Denmark, Aug. 5-9, 1996.