An Efficient Algorithm for Japanese Sentence Compaction Based on Phrase Importance and Inter-phrase Dependency

Rei Oguro[1], Kazuhiko Ozeki[1], Yujie Zhang[2], and Kazuyuki Takagi[1]

[1] The University of Electro-Communications, Chofu, Tokyo 182-8585, Japan
[2] ATR Spoken Language Translation Research Laboratories, Seika-cho, Kyoto 619-0288, Japan
{rei,ozeki,takagi}@ice.uec.ac.jp, yujie.zhang@slt.atr.co.jp

Abstract. This paper describes an efficient algorithm for Japanese sentence compaction. First, a measure of grammatical goodness of phrase sequences is defined on the basis of a Japanese dependency grammar. Also a measure of topical importance of phrase sequences is given. Then the problem of sentence compaction is formulated as an optimization problem of selecting a subsequence of phrases from the original sentence that maximizes the sum of the grammatical goodness and the topical importance. A recurrence equation is derived by using the principle of dynamic programming, which is then translated into an algorithm to solve the problem. The algorithm is of polynomial-time with respect to the original sentence length. Finally, an example of sentence compaction is presented.

1 Introduction

With the rapid increase of electronic publications, text summarization is becoming an important technical area in recent years[1]. Most widely used methods for text summarization are based on the idea of extracting important sentences from the whole text according to some criterion. However, in some applications such as on-line generation of TV closed-captions, it is sometimes necessary to shorten each sentence as it comes in. Requirements for such sentence compaction are:

1) Shortened sentences retain the original information as much as possible.
2) Shortened sentences are grammatically correct and natural.
3) The compaction rate is adjustable to meet the specification of the summarization system.
4) The computational load is small, and the computation can proceed without having to wait for the end of the input sentence.

Several methods for Japanese sentence compaction have been proposed [2, 3, 4, 5]. None of them, however, satisfies all the requirements above. In this paper, the sentence compaction problem is formulated as an optimization problem of selecting a subsequence of phrases from the original sentence that maximizes

the sum of grammatical goodness and topical importance. The grammatical goodness is defined on the basis of a Japanese dependency grammar. Then a recurrence equation is derived by using the principle of dynamic programming. This equation is easily translated into an algorithm, which is expected to satisfy all of the requirements 1) - 4). Finally, an example is presented to demonstrate how a sentence is shortened depending on the specified length.

2 Problem Formulation

Let $w_0 w_1 \cdots w_{M-1}$ be a Japanese sentence represented as a sequence of phrases, where a phrase is a syntactic unit called "bunsetsu" in Japanese, consisting of a content word with or without being followed by a string of function morphemes such as particles and auxiliary verbs. The problem of sentence compaction here is to select a "good" subsequence of phrases from the original sentence. In order to formulate this problem, a measure of goodness of a phrase sequence must be defined. The structure of a Japanese sentence can be viewed from a dependency grammatical point; it can be described by specifying which phrase modifies which phrase in a wide sense. Thus, a dependency structure on a subsequence $w_{k_0} w_{k_1} \cdots w_{k_{l-1}}$ of the original sentence can be represented by a mapping

$$c : \{k_0, k_1, \ldots, k_{l-2}\} \longrightarrow \{k_1, k_2, \ldots, k_{l-1}\} \tag{1}$$

such that $w_{c(m)}$ is the phrase modified by w_m $(m \in \{k_0, k_1, \ldots, k_{l-2}\})$. For a normal Japanese sentence, this mapping must satisfy

a) $m < c(m)$ $(m \in \{k_0, k_1, \ldots, k_{l-2}\})$,
b) if $m < n$ then $c(n) \leq c(m)$ $(m, n \in \{k_0, k_1, \ldots, k_{l-2}\})$.

The condition a) comes from the fact that modification always occurs from left to right in Japanese. The condition b) requires that two pairs of phrases in modification relation do not cross with each other.

Let $p(m, n)$ be a function that represents the strength of inter-phrase dependency between w_m and w_n, or the degree of validity for w_m to modify w_n. Then the grammatical goodness of the phrase sequence $w_{k_0} w_{k_1} \cdots w_{k_{l-1}}$ can be measured by $\max_c \sum_{i=0}^{l-2} p(k_i, c(k_i))$, where c runs over all the dependency structures on the phrase sequence. Let $q(m)$ be another function to represent the topical importance of w_m. Then the topical importance of the phrase sequence can be measured by $\sum_{i=0}^{l-1} q(k_i)$. The total goodness of the phrase sequence $w_{k_0} w_{k_1} \cdots w_{k_{l-1}}$ is defined as the sum of the grammatical goodness and the topical importance:

$$g(k_0, k_1, \ldots, k_{l-1}) \triangleq \begin{cases} q(k_0), & \text{if } l = 1; \\ \max_c \sum_{i=0}^{l-2} p(k_i, c(k_i)) + \sum_{i=0}^{l-1} q(k_i), & \text{otherwise.} \end{cases} \tag{2}$$

Now the problem of sentence compaction is formulated as follows:
Problem For a given sentence $w_0 w_1 \cdots w_{M-1}$ represented as a sequence of phrases, find its subsequence $w_{k_0} w_{k_1} \cdots w_{k_{N-1}}$ of a specified length N that maximizes the function $g(k_0, k_1, \ldots, k_{N-1})$.

3 Recurrence Equation

To solve the problem above, its partial solutions and the relationship among them are considered. Let $f(m,n,l)$ be the maximum value of $g(k_0,k_1,\ldots,k_{l-1})$ over all the subsequences of phrases of length l, starting with w_m and ending with w_n $(1 \le l \le n-m+1)$:

$$f(m,n,l) \stackrel{\triangle}{=} \max_{m=k_0<k_1<\ldots<k_{l-1}=n} g(k_0,k_1,\ldots,k_{l-1}) \tag{3}$$

Then it is shown that $f(m,n,l)$ satisfies the following recurrence equation:

1. for $m=n$ (only $l=1$ is allowed):

$$f(m,n,l) = q(m) \tag{4}$$

2. for $m<n$ $(2 \le l \le n-m+1)$:
   (a) for $l=2$:

$$f(m,n,l) = f(m,m,1) + f(n,n,1) + p(m,n) \tag{5}$$

   (b) for $l=3$:

$$f(m,n,l) = \max \begin{cases} f(m,m,1) + \max_{m<m'<n}\{f(m',n,2)\} + p(m,n); \\ \max_{m<n'<n}\{f(m,n',2) + p(n',n)\} + f(n,n,1) \end{cases} \tag{6}$$

   (c) for $l \ge 4$:

$$f(m,n,l) = \max \begin{cases} f(m,m,1) + \max_{m<m' \le n-l+2}\{f(m',n,l-1)\} + p(m,n); \\ \max_{2 \le l' \le l-2,\ m+l'-1 \le n' < m' \le n-l+l'+1} \\ \qquad \{f(m,n',l') + f(m',n,l-l') + p(n',n)\}; \\ \max_{m+l-2 \le n' < n}\{f(m,n',l-1) + p(n',n)\} + f(n,n,1) \end{cases} \tag{7}$$

This recurrence equation can be proved by noting that maximization of $g(k_0,k_1,\ldots,k_{l-1})$ with respect to $(k_0,k_1,\ldots,k_{l-1})$ subject to $m=k_0<k_1<\ldots<k_{l-1}=n$ is equivalent to maximization of $g(k_0,k_1,\ldots,k_{l-1})$ with respect to $(l',\ m',n',\ k_0,k_1,\ldots,k_{l'-1},\ k_{l'},k_{l'+1},\ldots,k_{l-1})$ subject to $1 \le l' \le l-1$, $m \le n' < m' \le n$, $m=k_0<k_1<\ldots<k_{l'-1}=n'$, $m'=k_{l'}<k_{l'+1}<\ldots<k_{l-1}=n$.

```
for n := 0 to M − 1 do
 begin
  for m := n downto 0 do
   begin if(m = n) then f(m, m, 1) := q(m);
    else
     begin
      for l := 2 to n − m + 1 do
       begin
        if(l = 2) then f(m, n, l) := Eq.(5);
        else if(l = 3) then f(m, n, l) := Eq.(6);
        else f(m, n, l) := Eq.(7);
       end;
     end;
   end;
 end;
```

Fig. 1. Algorithm for recursive computation of $f(m,n,l)$.

4 Algorithm

4.1 Tabular Computation

The above recurrence equation can be directly translated into an algorithm as shown in Fig.1. Because there is a large freedom in the order of computation, this is one of many versions of the translation that yield the same result. Actual computation is carried out by using a table to store the computed value of $f(m,n,l)$, which is looked up in later computation steps.

The time-complexity of this algorithm is of $O(M^6)$, where M is the length of the original sentence. Although $O(M^6)$ might seem to be large, there is no computational difficulty for practical values of M. In fact, the algorithm, coded in C, completes the computation for $M = 40$ within 1 sec on UltraSPARC-IIi (270MHz).

4.2 Back-Trace

In order to recover the optimum phrase sequence, it is necessary to keep the record of the values of variables, for back-pointers, that attain the maximum in the recurrence equation. Three tables are used for the purpose: $bp[m,n,l].lp$, $bp[m,n,l].np$, and $bp[m,n,l].mp$. In the middle case of Eq.(7), for example, the back-pointers are set as $bp[m,n,l].lp := l'$, $bp[m,n,l].np := n'$, and $bp[m,n,l].mp := m'$, where l', n', and m' are the values of those variables that attain the maximum. The back-trace proceeds in the following way. First, the optimum starting phrase and ending phrase are sought for a specified length N:

$$(m_0, n_0) := \mathrm{argmax}_{m,n} f(m,n,N). \tag{8}$$

Starting with (m_0, n_0) as the initial value, the function in Fig.2 performs a back-trace to give the final solution

$$solution := out(m_0, n_0, N). \tag{9}$$

```
function out(m, n, l): char string
  begin
    if (m = n) then out :="w_m";
    else
    begin
      l' := bp[m, n, l].lp;
      n' := bp[m, n, l].np;
      m' := bp[m, n, l].mp;
      out := out(m, n', l') ⊕ out(m', n, l − l');
    end;
  end.
```

Fig. 2. Back-trace function. The sign $\oplus$ denotes concatenation of two character strings.

5 Example of Sentence Compaction

An example of sentence compaction is presented in Table 1 to show how a sentence is shortened accoding to the specified length. The value of $q(m)$ was defined based on the part-of-speech of the content word in w_m. The value of $p(m,n)$ was defined on the basis of the morphological information about w_m and w_n, and also the inter-phrase distance between w_m and w_n in the original sentence[6]. As is seen in Table 1, all the shortened sentences are natural and make sense, retaining important information in the original sentence to the extent the length is allowed.

Original Sentence ($M = 9$): mata$^{(1)}$ (*also*) sodeya$^{(2)}$ (*sleeve and*) sodekuchi$^{(3)}$ (*end of sleeve*) pokettoguchinadoga$^{(4)}$ (*opening of pocket*) aburayogorede$^{(5)}$ (*by oil dirt*) henshokuwo$^{(6)}$ (*discolor*) okosu$^{(7)}$ (*be caused*) kotomo$^{(8)}$ (*cases*) arimasu$^{(9)}$ (*happen*).

(*Also some cases may happen that the sleeves, the end of sleeves, and the opening of pockets are discolored by oil dirt.*)

Table 1. Example of sentence compaction. Phrases are represented by their numbers in the original sentence. N denotes the specified length of the shortened sentence.

Length N	Shortened Sentence
8	(2) (3) (4) (5) (6) (7) (8) (9)
7	(2) (3) (4) (6) (7) (8) (9)
6	(2) (3) (4) (6) (7) (8)
5	(2) (3) (4) (6) (7)
4	(3) (4) (6) (7)
3	(6) (7) (8)
2	(6) (7)

6 Concluding Remarks

The Japanese sentence compaction problem was formulated as an optimization problem, and an efficient algorithm was derived. Because the computation proceeds input-synchronously, it is suited for on-line sentence compaction. The algorithm can also output the dependency structure (parse tree) of a shortened sentence in a manner similar to the one described in [7].

The values of $q(m)$ and $p(m,n)$ adopted in the example are only tentative. Therefore our future work includes:

1) Establishing a method to give the appropriate values of $q(m)$ and $p(m,n)$.
2) Finding an optimal weighting balance between the topical importance and the grammatical goodness.
3) Carrying out a large scale evaluation of this method.

References

1. Okumura, M., Nanba, H.: Automated text summarization: A survey. Journal of Natural Language Processing **6**(6)(1999) 1–26.
2. Wakao, T., Ehara, T., Shirai, K.: Summarization methods used for captions in TV news programs. Technical Report of Information Processing Society of Japan, 97-NL-122-13 (1997) 83–89.
3. Katoh, N., Uratani, N.: A new approach to acquiring linguistic knowledge for locally summarizing Japanese news sentences. Journal of Natural Language Processing **6**(7) (1999) 73–92.
4. Hori, C., Furui, S.: Automatic speech summarization based on word significance and linguistic likelihood. Technical Report of Information Processing Society of Japan, 99-SLP-29-18 (1999) 103–108.
5. Mikami, M., Masuyama, S., Nakagawa, S.: A summarization method by reducing redundancy of each sentence for making captions of newscasting. Journal of Natural Language Processing **6**(6) (1999) 65–81.
6. Zhang, Y., Ozeki, K.: Dependency analysis of Japanese sentences using the statistical property of dependency distance between phrases. Journal of Natural Language Processing **4**(2) (1997) 3–19.
7. Ozeki, K.: A multi-stage decision algorithm to select optimum bunsetsu sequences based on degree of kakariuke-dependency. IEICE Transactions **J70-D**(3) (1987) 601–609.

This article was processed using the LaTeX macro package with LLNCS style