Optimization for Efficient Determination of Chunk in Automatic Evaluation for Machine Translation

Hiroshi Echizen'ya[1] *Kenji Araki*[2] *Eduard Hovy*[3]

(1) Hokkai-Gakuen University, S 26-Jo, W 11-Chome, Chuo-ku, Sapporo 064-0926 Japan
(2) Hokkaido University, N 14-Jo, W 9-Chome, Kita-ku, Sapporo 060-0814 Japan
(3) Carnegie Mellon University, 5000 Forbes Avenue, Pittsburgh, PA 15213 USA
echi@lst.hokkai-s-u.ac.jp, araki@media.eng.hokudai.ac.jp, hovy@cmu.edu

Abstract

For automatic evaluation of machine translation, translation quality is commonly measuredly counting the number of chunks (consecutive words) that match between reference and candidate texts. For example, METEOR, GTM, ROUGE-W, and IMPACT all use chunks. The length and nature of chunks affects the measured result. IMPACT, a system that can determine the most suitable chunk, requires much processing time because it must scan a regular two-dimensional dynamic programming table. In this paper, we propose a new optimization method to determine chunks efficiently. Moreover, we developed a new automatic evaluation system using lemmas as lexical knowledge. We designate this system as **M**etric Using **L**emma and **O**ptimization for **C**hunk (MLOC). Through evaluation experiments, we confirm that the processing time of MLOC is shorter than that of IMPACT for reference and candidate text pairs with long sentences, although the lexical knowledge is used in MLOC, and MLOC indicates the highest correlation with human judgment among several automatic evaluation systems based on chunks.

Keywords: Automatic evaluation, Chunk, Dynamic programming, Tree structure, Lexical knowledge, Machine translation.

Proceedings of the First International Workshop on Optimization Techniques for Human Language Technology, pages 17–30,
COLING 2012, Mumbai, December 2012.

1 Introduction

In the field of machine translation, various methods for automatic evaluation have been proposed. The ULC(Giménez and Màrquez, 2007), MaxSim(Chan and Ng, 2008), and RTE(Padó et al., 2009) are known to produce high correlations with human judgment. Nevertheless, these methods are not widely used in machine translation because it is difficult to build the systems based on their methods. This problem might be solved when the systems are released on a web site. In contrast, BLEU(Papineni et al., 2002), NIST(NIST, 2002), PER(Su et al., 1992), and WER(Leusch et al., 2003) are used widely because we can build the systems easily based on their methods, and because their processing time is very short. However, these methods produce lower correlations with human judgment. Especially, the correlations assessed at the single-sentence level are quite low.

METEOR(Banerjee and Lavie., 2005), GTM(Turian et al., 2003), ROUGE-W(Lin and Och, 2004), and IMPACT(Echizen-ya and Araki, 2007), which are based on matching chunks between the reference and candidate texts, are effective for sentence-level correlation with human judgment. The chunk is a string of consecutive words considered for matching. The system IMPACT yields the highest correlations among various automatic evaluation methods in sentence-level correlations(Echizen-ya et al., 2009). IMPACT determines the most suitable chunk using information related to chunk length and position. However, IMPACT has a severe problem: it needs much process time to determine suitable chunks. This problem becomes an obstacle to realization of a practical automatic evaluation system using linguistic knowledge, although the quality of the system might improve when linguistic knowledge is used. Therefore, we propose an optimization method to determine the chunk efficiently in automatic evaluation based on chunks.

Our method generates a tree structure based on nodes corresponding to matching words. Consequently, it uses no dynamic programming table that includes information about non-matching words. Moreover, our method approximates the tree structure selecting the nodes which are important to determine suitable chunks. We developed a new automatic evaluation system based on our optimization method and use a lemma as lexical knowledge. We designate this system as **M**etric Using **L**emma and **O**ptimization for **C**hunk (MLOC). Our evaluation experiments indicate that the processing time of MLOC is shorter than that of IMPACT in reference and candidate text pairs containing long sentences, even though MLOC uses lemmas. Therefore, our optimization method is effective to decrease processing time. Moreover, we confirmed that MLOC provided the highest correlation with human judgments among several automatic evaluation systems based on chunks. Therefore, our optimization method is effective to develop a higher-quality and practical automatic evaluation system.

2 Problems of automatic evaluation method based on chunks

In METEOR, GTM, and ROUGE-W, no suitable chunk can be determined because the systems depend only on chunk length. However, IMPACT can determine suitable chunks using information about both chunk length and position. To do so, much processing time is needed.

For example, Table 1 shows a regular two-dimensional dynamic program table computing the Longest Common Subsequence (LCS) for two sequences as follows:

reference: glass guide of the plastic mounting panel P

candidate: a glass guide molded in panel member P made of resin

		j	1	2	3	4	5	6	7	8	9	10	11	12
		n_j	a	glass	guide	mold -ed	in	panel	mem -ber	P	made	of	the	resin
i	m_i	0	0	0	0	0	0	0	0	0	0	0	0	0
1	glass	0	0	**1**	1	1	1	1	1	1	1	2	2	2
2	guide	0	0	1	**2**	2	2	2	2	2	2	2	2	2
3	of	0	0	1	2	2	2	2	2	2	2	**3**	3	3
4	the	0	0	1	2	2	2	2	2	2	2	3	**4**	4
5	plas -tic	0	0	1	2	2	2	2	2	2	2	3	4	4
6	mount -ing	0	0	1	2	2	2	2	2	2	2	3	4	4
7	panel	0	0	1	2	2	2	**3**	3	3	3	3	4	4
8	P	0	0	1	2	2	2	3	3	**4**	4	4	4	4

Table 1: Example 1: A dynamic programming table.

In Table 1, the values (*i.e.*, i and j) outside of the table respectively denote the word positions in reference and candidate. The word number of the candidate is 12, which is the maximum value of j, and the word number of the reference is 8, which is the maximum value of i. The values in the table are obtained by Eq. (1).

$$D_{i,j} = \begin{cases} 0, & i = 0 \text{ or } j = 0 \\ max(D_{i-1,j}, D_{i,j-1}), & m_i \neq n_j \\ D_{i-1,j-1} + 1, & m_i = n_j \end{cases} \tag{1}$$

The length of LCS is 4 by Eq. (1) in Table 1. Moreover, the number of LCS routes is 2. The LCS routes are the sequences of matching words between the candidate and the reference. In Table 1, two LCS routes are obtained from the candidate and reference as shown below.

LCS route No.1
reference: [glass guide] of the plastic mounting [panel] [P]
candidate: a [glass guide] molded in [panel] member [P] made of the resin
LCS route No.2
reference: [glass guide] [of the] plastic mounting panel P
candidate: a [glass guide] molded in panel member P made [of the] resin

In LCS route No. 1, the matching words are "glass", "guide", "panel", and "P". In LCS route No. 2, the matching words are "glass", "guide", "of", and "the". These matching words correspond to bold values in Table 1. The consecutive matches form the chunk. Therefore, the number of chunks is 3 (*i.e.*, "glass guide", "panel", "P") in LCS route No. 1, and the number of chunks is 2 (*i.e.*, "glass guide", "of the") in LCS route No. 2. In this example, LCS route No. 1 must be selected because it has the most suitable chunk. In LCS route No. 2, "of the" in the candidate does not correspond to "of the" in the reference. Therefore, LCS route No. 1 has a more suitable chunk compared with LCS route No. 2.

In the determination of chunks, ROUGE-W and METEOR select LCS route No. 2, focusing only on chunk length. In LCS route No. 2, the quantities of words in two chunks ("glass guide", "of the") are 2. Moreover, in LCS route No. 1, the quantities of words in three chunks ("glass guide", "panel", "the") are, respectively, 2, 1, and 1. Therefore, METEOR selects LCS route No. 2, for which the number of chunks is small, because the penalty becomes a small value in the calculation of the score, and ROUGE-W also selects LCS route No. 2, for which the lengths of two chunks are 2.

In contrast, IMPACT selects LCS route No. 1 because it uses information of the position of the chunk, not only the chunk length. In LCS route No. 2, the positions of "of the" in candidate and reference are largely different. In IMPACT, the score is calculated using the length and position of chunk in each LCS route when several LCS routes are obtained following Eqs. (2) and (3). Only one LCS route, which has the highest score, is selected.

$$score = \sum_{c \in c\_num} \left(length(c)^{\beta} \times pos \right) \tag{2}$$

$$pos = \left(1.0 - \left| \frac{c_i}{m} - \frac{c_j}{n} \right| \right) \tag{3}$$

In Eq. (2), c means the chunks, and β is the weight parameter based on the length of each chunk ($\beta > 1.0$). The *pos* signifies the difference of the relative position of the chunk c between the candidate and reference. In Eq. (3), m and n respectively indicate the quantities of words in the candidate and reference. In addition, c_i and c_j respectively indicate the position of chunks of the candidate and reference. The score in LCS route No. 1 is $3.4933 (= 2^{1.2} \times (1.0 - |\frac{1}{8} - \frac{2}{12}|) + 1^{1.2} \times (1.0 - |\frac{7}{8} - \frac{6}{12}|) + 1^{1.2} \times (1.0 - |\frac{8}{8} - \frac{8}{12}|))$ when β is 1.2, and the score in LCS route No. 2 is $3.4461 (= 2^{1.2} \times (1.0 - |\frac{1}{8} - \frac{2}{12}|) + 2^{1.2} \times (1.0 - |\frac{3}{8} - \frac{10}{12}|))$. Therefore, IMPACT selects LCS route No. 1, whose score is higher than that of LCS route No. 2. As is clear, it is important for automatic evaluation based on chunks should use information about both the length and position of chunks.

However, IMPACT has a very severe problem. It requires much processing time to search all LCS routes to determine the LCS route which has the most suitable chunks. Moreover, in that case, IMPACT must scan all values of cells in the dynamic programming table although almost all values do not correspond to words that match between the candidate and reference texts. In ROUGE-W and METEOR, not much processing time is needed because they do not consider all LCS routes to determine the chunks. However, they cannot select the most suitable chunks. One way or the other, automatic evaluation systems based on chunks present severe problems. To solve this problem, we propose an optimization method to determine the most suitable chunk efficiently.

3 Optimization method for chunk determination

3.1 Mapping from the dynamic programming table to the tree structure

Table 2 shows an example of the dynamic programming table generated to obtain matching words for the following the candidate and reference fragments:

reference: array rules determine the limits to designing wiring routes
candidate: for arrangement of restriction on the design rule, the wiring route is determined

	j	1	2	3	4	5	6	7	8	9	10	11	12	13
	n_j	ar -range -ment	of	re -stric -tion	on	the	de -sign	rule	,	the	wir -ing	route	is	de -ter -mined
i m_i	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1 array	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2 rules	0	0	0	0	0	0	0	0	0	0	0	0	0	0
3 de -ter -mine	0	0	0	0	0	0	0	0	0	0	0	0	0	0
4 the	0	0	0	0	0	$\underline{\mathbf{1}}$	1	1	1	$\underline{\mathbf{1}}$	1	1	1	1
5 limit	0	0	0	0	0	1	1	1	1	1	1	1	1	1
6 to	0	0	0	0	0	1	1	1	1	1	1	1	1	1
7 de -sign -ing	0	0	0	0	0	1	1	1	1	1	1	1	1	1
8 of	0	0	$\underline{\mathbf{1}}$	1	1	1	1	1	1	1	1	1	1	1
9 the	0	0	1	1	1	$\underline{\mathbf{2}}$	2	2	2	$\underline{\mathbf{2}}$	2	2	2	2
10 wir -ing	0	0	1	1	1	2	2	2	2	3	$\underline{\mathbf{3}}$	3	3	3
11 routes	0	0	1	1	1	2	2	2	2	3	3	3	3	3

Table 2: Example 2: Dynamic programming table.

In IMPACT, LCS routes of three kinds are obtained from the dynamic programming table of Table 2, scanning all values of cells as described below.

LCS route No.1
reference: array rules determine the limit to designing [of] [the wiring] routes
candidate: arrangement [of] restriction on the design rule , [the wiring] route is determined
LCS route No.2
reference: array rules determine [the] limit to designing of [the wiring] routes
candidate: arrangement of restriction on [the] design rule , [the wiring] route is determined
LCS route No.3
reference: array rules determine the limit to designing [of] [the] [wiring] routes
candidate: arrangement [of] restriction on [the] design rule , the [wiring] route is determined

In the example given above, the LCS routes with the most suitable chunks are LCS route No. 1 and No. 2. LCS route No. 2 is selected in IMPACT by the information of the position of chunk "the". However, IMPACT needs much processing time because it must scan all values of cells that do not correspond to the matching words in Table 2.

Our method replaces the information of the matching words in a dynamic programming table with a tree structure. Therefore, our method can emphasize matching words without scanning the values of cells that do not correspond to the matching words in the dynamic programming table. This means that our method can decrease processing time.

First, our method generates the dynamic programming table and extracts only $D_{[i,j]}$ which correspond to the matching words. In Table 2, the values shown in bold typeface correspond to the matching words. Therefore, all $D_{[i,j]}$ of the matching words are $D_{[4,5]}$, $D_{[4,9]}$, $D_{[8,2]}$, $D_{[9,5]}$, $D_{[9,9]}$ and $D_{[10,10]}$. Our method requires no dynamic programming table after extraction of the matching words. Moreover, our method sorts $D_{[i,j]}$ from large to small values. Each $D_{[i,j]}$ corresponds to a node of the tree structure. Figure 1 presents an example of the sort of $D_{[i,j]}$.

value	$D_{[i,j]}$
3	$D_{[10,10]}$
2	$D_{[9,9]}$ $D_{[9,5]}$
1	$D_{[8,2]}$ $D_{[4,5]}$ $D_{[4,9]}$

Figure 1: Example of the sorting of $D_{[i,j]}$.

Our method generates a tree structure linking two $D_{[i,j]}$. Figure 2 shows the algorithm for linking between two nodes in a tree structure.

```
Start:
level_num = # of level in value
for(s = level_num; s > 1; s--)
    num_1 = # of $D_{[i,j]}^{s}$
    for(t = 1; t > num_1; t++)
        num_2 = # of $D_{[i,j]}^{s-1}$
        for(u = 1; u > num_2; u++)
            If $i$ in $D_{[i,j]}^{t}$ > $i$ in $D_{[i,j]}^{u}$ and $i$ in $D_{[i,j]}^{t}$ > $j$ in $D_{[i,j]}^{u}$
                $D_{[i,j]}^{t}$ -> $D_{[i,j]}^{u}$    # generation of link
End:
```

Figure 2: Algorithm for a link in the tree structure.

In Fig. 2, the value of *level_num* is 3 because the level of values of $D_{[i,j]}$ is 1–3. Moreover, the value of *num*_1 is 1 because $D_{[i,j]}$ in the level of value 3 is only $D_{[10,10]}$ and the value of *num*_2 is 2 because $D_{[i,j]}$ in level of value 2 is $D_{[9,9]}$ and $D_{[9,5]}$. Our method compares $D_{[10,10]}$ with $D_{[9,9]}$. As a result, the link is generated because $i = 10$ in $D_{[10,10]}$ is larger than $i = 9$ in $D_{[9,9]}$ and $j = 10$ in $D_{[10,10]}$ is larger than $j = 9$ in $D_{[9,9]}$. Therefore, $D_{[10,10]}$ and $D_{[9,9]}$ can be connected as the matching word sequence. In also $D_{[10,10]}$ and $D_{[9,5]}$, the link is generated because $i = 10$ in $D_{[10,10]}$ is larger than $i = 9$ in $D_{[9,5]}$, and $j = 10$ in $D_{[10,10]}$ is larger than $j = 5$ in $D_{[9,5]}$.

Moreover, our method specifically examines the level of values 2 and 1. As a result, for $D_{[9,5]}$ and $D_{[4,5]}$, the link is not generated because $j = 5$ in $D_{[9,5]}$ is not larger than $j = 5$ in $D_{[4,5]}$. In $D_{[9,5]}$ and $D_{[4,9]}$, the link is not generated because $j = 5$ in $D_{[9,5]}$ is not larger than $j = 9$ in $D_{[4,9]}$. Therefore, $D_{[9,5]}$ is not connected with $D_{[4,5]}$ and $D_{[4,9]}$ as the matching word sequence. Finally, the $D_{[4,9]}$ is deleted as the node because it has no link with other nodes. Figure 3 depicts the tree structure using the algorithm presented in Fig. 2. Our method does not require the scanning of those (many) cells that do not correspond to the matching words in the dynamic programming table. Therefore, our method can decrease the processing time.

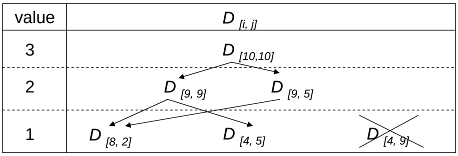


Figure 3: Example of the tree structure.

3.2 Approximation in tree structure

Our method approximates the tree structure using only nodes which are important from the perspective of the length and position of chunks to decrease the processing time. Figure 4 shows the algorithm for selection of nodes in a tree structure.

In our method, the nodes selected are the ones that continue with other nodes, and those for which the difference of relative position between i and j is the smallest among all nodes at one level of value. This means that our method effectively uses both length and position of chunks. In the example of the tree structure in Fig. 3, $D_{[10,10]}$, which is the node at the top level, and $D_{[8,2]}$ and $D_{[4,5]}$, which are the nodes at a lower level, are selected first. Next, our method selects only the most important node among $D_{[9,9]}$ and $D_{[9,5]}$ of level 2.

By Fig. 4, the value of num_1 is 2 (*i.e.*, $D_{[9,9]}$ and $D_{[9,5]}$), and the value of num_2 is 2 (*i.e*, $D_{[8,2]}$ and $D_{[4,5]}$). Therefore, our method compares $D_{[9,9]}$ respectively with $D_{[8,2]}$, $D_{[4,5]}$, and $D_{[9,5]}$ with $D_{[8,2]}$. As a result, neither $D_{[9,9]}$ nor $D_{[9,5]}$ is selected as a node at this time because these nodes are not mutually continuous with $D_{[8,2]}$ and $D_{[4,5]}$. Moreover, the value of num_3 is 1 (*i.e.*, $D_{[10,10]}$). Therefore, our method compares respectively $D_{[9,9]}$ with $D_{[10,10]}$, and $D_{[9,5]}$ with $D_{[10,10]}$. Results show that $D_{[9,9]}$ is selected as a node because $i = 9$ in $D_{[9,9]}$ becomes $i = 10$ in $D_{[10,10]}$ by adding 1 and $j = 9$ in $D_{[9,9]}$ also becomes $j = 10$ in $D_{[10,10]}$ by adding 1: $D_{[9,9]}$ is continuous with $D_{[10,10]}$.

Moreover, our method selects the node for which the difference of word position is the smallest. In this example, the value of differences of word position are, respectively 0.1259 ($=|\frac{9}{11} - \frac{9}{13}|$) in $D_{[9,9]}$ and 0.4336 ($=|\frac{9}{11} - \frac{5}{13}|$) in $D_{[9,5]}$. Therefore, only $D_{[9,9]}$ is selected as the node of tree structure, and $D_{[9,5]}$, which does not continue with other nodes and the difference of word position is not the smallest, is deleted from the tree structure. Figure 5 depicts the final tree structure. Our method obtains the LCS route using only a final tree structure.

```
Start:
        level_num = # of level in value
        for(s = level_num-1; s > 1; s--)
            num_1 = # of $D_{[i,j]}^{s}$
LOOP:       for(t = 1; t > num_1; t++)
                num_2 = # of $D_{[i,j]}^{s-1}$
                for(u = 1; u > num_2; u++)
                    If $i$ in $D_{[i, j]}$ $^{t}$-1== $i$ in $D_{[i, j]}^{u}$ and $i$ in $D_{[i, j]}^{t}$-1 == $j$ in $D_{[i, j]}^{u}$
                        tree.node($D_{[i, j]}^{t}$ )   # addition node to tree
                        go to LOOP
                num_3 = # of $D_{[i,j]}^{s+1}$
                for(v = 1; v > num_3; v++)
                    If $i$ in $D_{[i, j]}$ $^{t}$+1== $i$ in $D_{[i, j]}^{v}$ and $i$ in $D_{[i, j]}^{t}$+1 == $j$ in $D_{[i, j]}^{v}$
                        tree.node($D_{[i, j]}^{t}$ )   # addition node to tree
                        go to LOOP
            $min\_D_{[i, j]}$ = min_diff($D_{[i,j]}^{s}$)
            tree.node($min\_D_{[i, j]}$ )   # addition node to tree
End:
```

Figure 4: Algorithm for selection of nodes in a tree structure.

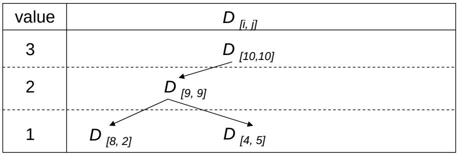


Figure 5: Example of the final tree structure.

As a result, our method can delete LCS routes that have no suitable chunk (*i.e.*, LCS route No. 3 described in Section 3.1): our method can decrease processing time using only necessary $D_{[i,j]}$.

4 MLOC: Automatic evaluation system using lemma

We developed a new automatic evaluation system for machine translation. Our system uses the optimization method described in Section 3. Moreover, it uses a lemma as lexical information to increase the matching words. We designate this system as **M**etric Using **L**emma and **O**ptimization for **C**hunk (MLOC). Consider the following example, using lemmas.

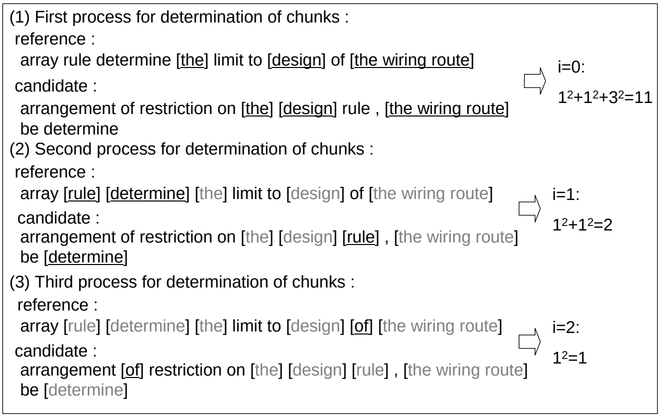
reference: array rule determine the limit to design of the wiring route
candidate: arrangement of restriction on the design rule, the wiring route be determine

MLOC calculates scores using all chunks. To do so, it uses the following Eqs. (4) and (5).

$$R = \left(\frac{\sum_{i=0}^{RN} \left(\alpha^i \sum_{c \in c\_num} length(c)^\beta \right)}{m^\beta} \right)^{\frac{1}{\beta}} \tag{4}$$

$$P = \left(\frac{\sum_{i=0}^{RN} \left(\alpha^i \sum_{c \in c\_num} length(c)^\beta \right)}{n^\beta} \right)^{\frac{1}{\beta}} \tag{5}$$

Eqs. (4) and (5) respectively show the recall and precision. Figure 6 presents an example of determination of all chunks between the reference and candidate.



(1) First process for determination of chunks :

reference :

array rule determine [the] limit to [design] of [the wiring route]

candidate :

arrangement of restriction on [the] [design] rule , [the wiring route] be determine

i=0: $1^2+1^2+3^2=11$

(2) Second process for determination of chunks :

reference :

array [rule] [determine] [the] limit to [design] of [the wiring route]

candidate :

arrangement of restriction on [the] [design] [rule] , [the wiring route] be [determine]

i=1: $1^2+1^2=2$

(3) Third process for determination of chunks :

reference :

array [rule] [determine] [the] limit to [design] [of] [the wiring route]

candidate :

arrangement [of] restriction on [the] [design] [rule] , [the wiring route] be [determine]

i=2: $1^2=1$

Figure 6: Example of determination of all chunks.

MLOC determines only one LCS route using the process described in Section 3. Moreover, LCS routes are determined recursively for all matching words. In Fig. 6, "the", "design", and "the wiring route" are selected as chunks in the first process for determination of chunks. Therefore, the value of $\sum_{c \in c\_num} length(c)^\beta$ is $11(=1^{2.0}+1^{2.0}+3^{2.0})$ when β is 2.0. In the second process for determination of chunks, "rule" and "determine" are selected as chunks. The value of $\sum_{c \in c\_num} length(c)^\beta$ is $2(=1^{2.0}+1^{2.0})$. Finally, "of" is selected as a chunk in the third process for chunk determination. The value of $\sum_{c \in c\_num} length(c)^\beta$ is $1(=1^{2.0})$. Therefore, the value of the Reputation number for determination of suitable LCS (*i.e.*, RN) in Eqs. (4) and (5) becomes 2. The value of $\sum_{i=0}^{RN} \left(\alpha^i \sum_{c \in c\_num} length(c)^\beta \right)$ is $12.25(=0.5^0 \times 11 + 0.5^1 \times 1 + 0.5^2 \times 1)$ when α is 0.5. Moreover, the values of R and P in Eqs. (4) and (5) are, respectively 0.3182 $(=\sqrt{\frac{12.25}{11^{2.0}}})$ and 0.2692 $(=\sqrt{\frac{12.25}{13^{2.0}}})$. MLOC obtains the final score by Eq. (6) as

$$score = \frac{(1+\gamma^2)RP}{R+\gamma^2 P}. \quad (6)$$

In Eq. (6), γ is determined as P/R. The value of γ is 0.8460 when R is 0.2692 and P is 0.3182. The final score is $0.2877(=\frac{(1+0.8460^2)\times 0.3182\times 0.2692}{0.3182+0.8460^2\times 0.2692})$ by Eq. (6). MLOC can obtain many matching words using lemmas. Moreover, it can obtain the score efficiently determining the suitable LCS route using the optimization process.

5 Experiments

5.1 Experimental Procedure

References and candidates were obtained from patent data in NTCIR-7(Fujii et al., 2008). Using these data, 14 machine translation systems translated 100 Japanese sentences into 100 English sentences. Therefore, the number of candidates is 1,400 (=14×100). Moreover, four references are given to each candidate obtained by each machine translation system. First, we compared MLOC, our system using only the optimization process (MLOC without lemma) and IMPACT from the perspective of the process time.

Moreover, we calculated the correlation between the scores by the automatic evaluation systems and scores by human judgment. In the scores by human judgment, the human judge evaluated 1,400 candidates from the perspective of adequacy and fluency on a scale of 1 to 5. We used the median value in the evaluation results of three human judges as the final scores of 1–5. We calculated the Pearson's correlation coefficient and Spearman's rank correlation coefficient between the scores obtained using the automatic evaluation systems and the scores by human judgments at the sentence-level and system-level. In the automatic evaluation systems for machine translation, three systems (*i.e.*, MLOC, IMPACT, and ROUGE-W, which are systems based on the chunk) are used. IMPACT especially indicated high correlation coefficients in NTCIR-7 data(Echizen-ya et al., 2009). In ROUGE-W, 1.2 was used as the value of weight parameter for the length of chunk in preliminary experiments. In MLOC and IMPACT, 0.1 was used as the value of penalty parameter (*i.e.*, α) for the difference of chunk sequence and 1.2 was used as the value of weight parameter (*i.e.*, β) for the length of chunks in Eqs. (4) and (5). The MLOC using lemma replaced all words with the lemma in all references and candidates using output obtained by TreeTagger(Schmid, 1994).

5.2 Experimental Results

	all 1,400 pairs	selected 8 pairs
MLOC	202 sec	6 sec
MLOC without lemma	144 sec	5 sec
IMPACT	149 sec	7 sec

Table 3: Processing time.

In Table 3, the processing time of MLOC, MLOC without lemmas, and IMPACT were respectively 202 sec, 144 sec, and 149 sec using all 1,400 pairs of four references and candidates. The processing time of MLOC was the longest because of the use of lemma in creased the number of matches. The processing time of MLOC without lemmas is only 5 sec lower than that of IMPACT. The reason is that the number of LCS routes between the references and candidates are almost invariably small: less than 100. Our optimization method exhibits its strength for cases where the number of LCS routes are large. Therefore, we selected eight pairs of four reference and candidate sentences with more than 300 LCS routes as determined by IMPACT. For this test, IMPACT required 7 sec to process the eight pairs of four references and candidates. However, the processing time was 5 sec for MLOC without lemmas, and 6 sec for MLOC with lemmas. Here, even with the use of lemmas, the processing time of MLOC was shorter than that of IMPACT. These results indicate that our optimization method is effective for decreasing the processing time in the sentences which the number of LCS routes is large.

Table 4 and Table 5 respectively portray Pearson's correlation coefficient in adequacy and fluency. Table 6 and Table 7 respectively portray Spearman's rank correlation coefficient in adequacy and fluency.

	No. 1	No. 2	No. 3	No. 4	No. 5	No. 6	No. 7	No. 8	No. 9	No. 10	No. 11	No. 12	No. 13	No. 14	Avg.	System
MLOC	**0.7770**	**0.5405**	**0.4821**	0.5680	0.5477	**0.6310**	**0.6800**	0.7348	0.7058	0.5691	0.7007	**0.6490**	0.7669	**0.5488**	**0.6358**	0.9271
MLOC without lemma	0.7625	0.5307	0.4704	0.5566	**0.5518**	0.6295	0.6516	**0.7375**	**0.7113**	0.5813	0.7095	0.6251	0.7677	0.5321	0.6298	0.9266
IMPACT	0.7625	0.5307	0.4704	0.5566	**0.5518**	0.6295	0.6516	0.7374	**0.7113**	0.5813	0.7097	0.6251	**0.7685**	0.5321	0.6299	0.9264
ROUGE-W	0.7648	0.5044	0.4615	**0.5765**	0.5482	0.6257	0.6415	0.7336	0.7072	**0.5938**	**0.7131**	0.6099	0.7643	0.5402	0.6275	**0.9400**

Table 4: Pearson's correlation coefficient in adequacy.

	No. 1	No. 2	No. 3	No. 4	No. 5	No. 6	No. 7	No. 8	No. 9	No. 10	No. 11	No. 12	No. 13	No. 14	Avg.	System
MLOC	**0.5768**	**0.3880**	**0.3999**	**0.5857**	0.4728	0.5630	**0.5383**	**0.7112**	0.5517	0.5243	0.6272	**0.3803**	**0.6129**	0.3897	**0.5223**	0.9382
MLOC without lemma	0.5543	0.3765	0.3705	0.5548	**0.4737**	0.5786	0.5168	0.6968	**0.5564**	0.5514	0.6333	0.3727	0.6081	0.4012	0.5175	0.9382
IMPACT	0.5543	0.3765	0.3705	0.5548	**0.4737**	0.5786	0.5168	0.6968	**0.5564**	0.5514	0.6333	0.3728	0.6081	0.4012	0.5175	0.9381
ROUGE-W	0.5566	0.3501	0.3504	0.5715	0.4693	**0.5791**	0.5006	0.6941	0.5480	**0.5520**	**0.6410**	0.3568	0.6003	**0.4078**	0.5127	**0.9426**

Table 5: Pearson's correlation coefficient in fluency.

In Tables 4–7, Nos. 1–14 denote the respective machine translation systems in NTICR-

	No. 1	No. 2	No. 3	No. 4	No. 5	No. 6	No. 7	No. 8
MLOC	**0.7513**	0.4609	**0.5196**	0.5738	**0.4902**	**0.6395**	**0.6530**	**0.6539**
MLOC with -out lemma	0.7468	**0.4618**	0.4923	0.5666	0.4877	0.6280	0.6196	0.6464
IMPACT	0.7468	**0.4618**	0.4923	0.5666	0.4877	0.6280	0.6196	0.6460
ROUGE-W	0.7379	0.4494	0.4943	**0.5786**	0.4785	0.6166	0.5902	0.6375
	No. 9	No. 10	No. 11	No. 12	No. 13	No. 14	Avg.	System
MLOC	**0.6882**	0.5510	0.6982	**0.6195**	0.7439	**0.5864**	**0.6164**	0.9824
MLOC with -out lemma	0.6859	0.5478	0.7171	0.5960	0.7433	0.5836	0.6088	**0.9912**
IMPACT	0.6859	0.5478	0.7171	0.5971	0.7442	0.5836	0.6089	**0.9912**
ROUGE-W	0.6734	**0.5653**	**0.7195**	0.5737	**0.7448**	0.5682	0.6020	**0.9912**

Table 6: Spearman's rank correlation coefficient in adequacy.

	No. 1	No. 2	No. 3	No. 4	No. 5	No. 6	No. 7	No. 8
MLOC	**0.5715**	**0.3732**	**0.3799**	**0.5829**	0.4101	**0.6169**	**0.4880**	**0.6590**
MLOC with -out lemma	0.5520	0.3518	0.3643	0.5488	**0.4119**	0.5995	0.4691	0.6321
IMPACT	0.5520	0.3518	0.3643	0.5488	**0.4119**	0.5995	0.4691	0.6325
ROUGE-W	0.5426	0.3359	0.3472	0.5482	0.4066	0.5898	0.4497	0.6275
	No. 9	No. 10	No. 11	No. 12	No. 13	No. 14	Avg.	System
MLOC	0.5436	0.4339	0.6469	**0.3689**	0.6363	0.4117	**0.5089**	0.9165
MLOC with -out lemma	**0.5452**	0.4661	0.6548	0.3590	**0.6371**	**0.4437**	0.5025	**0.9253**
IMPACT	**0.5452**	0.4661	0.6544	0.3586	0.6359	**0.4437**	0.5024	**0.9253**
ROUGE-W	0.5277	**0.4738**	**0.6686**	0.3423	0.6283	0.4241	0.4937	**0.9253**

Table 7: Spearman's rank correlation coefficient in fluency.

7. Their values are correlation coefficients at the sentence level. In the tables, "Avg." signifies the average values of 14 correlation coefficients. "System" stands for the correlation coefficients at the system level. Bold values show the highest correlation coefficient among four automatic evaluation systems.

5.3 Discussion

We describe the influence of the optimization method from the perspective of the scores. In Tables 4–7, the difference between MLOC without lemmas and IMPACT depends on whether the optimization method described in Section 3 is used or not. Therefore, in MLOC without lemmas and IMPACT of Tables 4–7, we can confirm the influence of the optimization method on scores. Results show that the influence of the optimization method is very slight because the difference of values for the correlation coefficient of "Avg." and "System" between MLOC without lemmas and IMPACT is 0.0001 or 0.0002 in Tables 4–7. The scores of MLOC without lemmas are almost identical to the scores of IMPACT. Therefore, we confirm that the influence of the optimization method is very slight on scores.

In the correlation coefficient, MLOC indicated the highest values among four systems. Table 8 presents the average values of "Avg." and "System" in Tables 4–7. In Table 8, the total average value of correlation coefficient of MLOC is the highest among four systems. The reason is that MLOC is extremely effective at the sentence level. The numbers for which MLOC obtained the highest correlation coefficient among four systems in 14 machine translation systems (*i.e.*, the quantities of bold values in MLOC) are, respectively, 7, 8, 9, and 8 in Tables 4, 5, 6 and 7. These results show that MLOC is extremely effective for sentence-level evaluation.

	Pearson in adequacy	Pearson in fluency	Spearman in adequacy	Spearman in fluency	Avg.
MLOC	0.7815	0.7303	0.7994	0.7127	**0.7560**
MLOC with -out lemma	0.7782	0.7279	0.8000	0.7139	0.7550
IMPACT	0.7781	0.7278	0.8001	0.7138	0.7550
ROUGE-W	0.7838	0.7277	0.7966	0.7095	0.7544

Table 8: Average values of "Avg." and "System".

6 Conclusion

In this paper we propose an optimization method for the efficient determination of chunks for automatic evaluation of machine translation output using chunk-based matching, and developed MLOC as a new automatic evaluation system. The experimentally obtained results demonstrate that the processing time of MLOC is shorter than that of IMPACT for the pairs of reference and candidate fragments in which the number of LCS route is more than 300, even though lemmas are used in MLOC. These results indicate that our optimization method is effective to decrease the processing time.

Moreover, we performed experiments to confirm the quality of MLOC from the perspective of the automatic evaluation system. Our evaluation experiments confirm that MLOC can obtain the highest correlation coefficients among other systems based on chunks. This means that our optimization method is effective to realize higher quality automatic evaluation for machine translation.

Future studies will use a part-of-speech as effective lexical knowledge to improve MLOC quality. Moreover, we plan to use MLOC for parameter tuning in SMT.

Acknowledgments

This work was done as research under the AAMT/JAPIO Special Interest Group on Patent Translation. The Japan Patent Information Organization (JAPIO) and the National Institute of Information (NII) provided corpora used in this work. The author gratefully acknowledges support from JAPIO and NII. Also, this work was partially supported by Grants from the High-Tech Research Center of Hokkai-Gakuen University.

References

Jesús Giménez and Lluís Màrquez. (2007). Heterogeneous Automatic MT Evaluation Through Non-Parametric Metric Combinations, *In Proceedings of the IJCNLP'07*, pp.319–326.

Yee Seng Chan and Hwee Tou Ng. (2008). MaxSim: An Automatic Metric for Machine Translation Evaluation Based on Maximum Similarity, *In Proceedings of the MetricMATR Workshop of AMTA-2008*, pp.319–326.

Sebastian Padó, Michel Galley, Dan Jurafsky and Christopher D. Manning. (2009). Textual Entailment Features for Machine Translation Evaluation, *In Proceedings of the Fourth Workshop on Statistical Machine Translation*

Kishore Papineni, Salim Roukos, Todd Ward and Wei-Jing Zhu. 2002. Bleu: a Method for Automatic Evaluation of Machine Translation. *In Proc. of ACL'02*, pp.311–318.

NIST. 2002. Automatic Evaluation of Machine Translation Quality Using N-gram Co-Occurrence Statistics. *http://www.nist.gov/speech/tests/mt/doc/ngram-study.pdf.*

Keh-Yih Su, Ming-Wen Wu and Jing-Shin Chang. 1992. A New Quantitative Quality Measure for Machine Translation Systems. *In Proc. of GOLING'92*, pp.433–439.

Gregor Leusch, Nicola Ueffing and Hermann Ney. 2003. A Novel String-to-String Distance Measure with Applications to Machine Translation Evaluation. *In Proc. of MT Summit IX*, pp.240–247.

Satanjeev Banerjee and Alon Lavie. 2005. METEOR: An Automatic Metric for MT Evaluation with Improved Correlation with Human Judgments. *In Proc. of ACL Workshop on Intrinsic and Extrinsic Evaluation Measures for Machine Translation and/or Summarization*, pp.65–72.

Joseph P. Turian, Luke Shen and I. Dan Melamed. 2003. Evaluation of Machine Translation and its Evaluation. *In Proc. of MT Summit IX*, pp.386–393.

Chin-Yew Lin and Franz Josef Och. (2004). Automatic Evaluation of Machine Translation Quality Using Longest Common Subsequence and Skip-Bigram Statistics, *In Proceedings of the ACL'04*, pp.606–613.

Hiroshi Echizen-ya and Kenji Araki. 2007. Automatic Evaluation of Machine Translation based on Recursive Acquisition of an Intuitive Common Parts Continuum. *In Proc. of MT Summit XII*, pp.151–158.

Hiroshi Echizen-ya, Terumasa Ehara, Sayori Shimohata, Atsushi Fujii, Masao Utiyama, Mikio Yamamoto, Takehito Utsuro and Noriko Kando. 2009. Meta-Evaluation of Automatic Evaluation Methods for Machine Translation using Patent Translation Data in NTCIR-7. *In Proc. of the Third Workshop on Patent Translation*, pp.9–16.

Atsushi Fujii, Masao Utiyama, Mikio Yamamoto and Takehito Utsuro. 2008. Overview of the Patent Translation Task at the NTCIR-7 Workshop. *In Proc. of Seventh NTCIR Workshop Meeting on Evaluation of Information Access Technologies: Information Retrieval, Question Answering and Cross-lingual Information Access*, pp.389–400.

Helmut Schmid. 1994. Probabilistic Part-of-Speech Tagging Using Decision Trees. *In Proc. of International Conference on New Methods in Language Processing* pp.389–400.