EBMT Based on Finite Automata State Transfer Generation

Ren Feiliang, Zhang Li, Hu Minghan, Yao Tianshun
NLP Laboratory, Institute of Computer Software and Theory,
Northeastern University, Shenyang, 110004, China
renfeiliang@ise.neu.edu.cn

Abstract

This paper proposes an EBMT method based on finite automata state transfer generation. In this method, first some links from the fragments in the input sentence to the fragments in the target sentence of the selected example are built. Then some predefined states are assigned to these links according to their link types. Finally, taking these links and their corresponding states as inputs, a finite automaton is constructed and the translation result is generated in a finite automata state transfer manner. This method can be easily replicated, and does not need too much complicated parsers either. Based on this method, we built a Chinese-Japanese bidirectional EBMT system to evaluate the proposed method, and experimental results indicate that the proposed method is effective.

1 Introduction

Example-based machine translation (EBMT) is a method of translation by the principle of analogy. It generally consists of three modules: a matching module, an alignment module and a recombination module. Given an input sentence, an EBMT system first matches the input sentence against the example set to select some relevant examples whose source sentence parts are similar to the given input sentence; once the relevant examples have been selected, the alignment module will select the corresponding fragments in the target sentences of the selected examples for every part of the input sentence; once the appropriate fragments have been selected, the recombination module will combine them to form a legal target text (Somers, 1999).

Generally, we can regard the last two modules as a translation generation module. For the generation, some researchers (Aramaki and Kurohashi, 2003; Aramaki and Kurohashi, 2004) used a semantic-based generation approach that obtains an appropriate translation fragment for each part of the input sentence. The final translation is generated by recombining the translation fragments in some order. This approach does not take into account the fluency between the translation fragments. The statistical approach (Akiba et al., 2002; Watanabe and Sumita, 2003; Imamura et al., 2004) selects translation fragments with a statistical model. The statistical model can improve the fluency between the translation fragments by using *n*-gram co-occurrence statistics. However, the statistical model does not take into account the semantic relation between the example and the input sentence. Tree parsing based generation approach (Zhanyi et al., 2005) solves the above two problems by using a method based on tree string correspondence (TSC) and statistical generation. During the translation process of this method, the input sentence is first parsed into a tree. Then the TSC forest is searched to find out if it is best matched with the parse tree. Finally, it uses a statistical generation model to generate translation by combining the target language strings in the TSCs. This method depends heavily on the tree parsing technology, if the parser does not work well, it is impossible to generate a proper translation result.

This paper proposes a generation method for EBMT based on finite automata state transfer. It uses the target sentence of the selected example to generate the translation result in a finite auto-

mata state transfer manner, and outputs the modified target sentence as final translation result.

The rest of this paper is organized as follows. Section 2 gives a brief description of our Chinese-Japanese bidirectional EBMT system. Section 3 describes our generation method in detail. Section 4 presents our experiments. At last, we conclude this paper and present future work in section 5.

2 System Structure of Our Chinese-Japanese Bidirectional EBMT System

Our Chinese-Japanese bidirectional EBMT system's structure is shown in figure 1. A word-based matching method is used to select one example that is most similar to the input sentence. Here two sentences' similarity is calculated as shown in formula 1 (LV Xue-qiang and Ren Feiliang, 2003).

$$Sim(s_1, s_2) = \frac{2 \times SameWord(s_1, s_2)}{Len(s_1) + Len(s_2)} \qquad (1)$$

In this formula, $Sim(s_1, s_2)$ means the similarity of sentence s_1 and sentence s_2, $SameWord(s_1, s_2)$ means the number of common words in sentence s_1 and sentence s_2, and $Len(s_i)$ is the number of total words in sentence s_i.

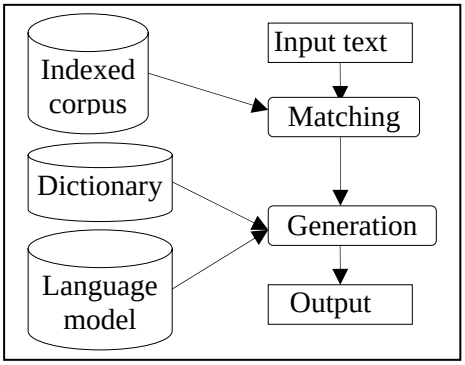

Figure 1. Structure of CJ EBMT System

3 Generation Based on Finite Automata State Transfer

We generate the input sentence's translation by modifying the target sentence of the selected example. This process consists of three steps.

(1) Build links from the fragments in the input sentence to the fragments in the target sentence of the selected example.
(2) Assign states to each of these links.
(3) Construct a finite automaton and generate the translation result in a automaton state transfer manner.

3.1 Building Links

A link from a fragment in one sentence S_1 to a fragment in another sentence S_2 is defined as a 3-tuple (Sf_i, Tf_j, t), where Sf_i (a fragment in S_1), Tf_j (a fragment in S_2), and t are called source fragment, target fragment, and link type respectively. In this 3-tuple, if the languages of S_1 and S_2 are the same, the target fragment is the most similar part in S_2 to the source fragment; if the languages of S_1 and S_2 are different, the target fragment is the most useful part in S_2 to generate the source fragment's translation. Either the source fragment or the target fragment can be null, but they can't be null at the same time. Link type indicates a possible operation converting the source fragment to the target fragment. Following edit distance's style (Wagner and Fischer, 1974), we define four link types: *I, R, D, N*, which mean *inserting, replacing, deleting* and *outputting directly* respectively.

Suppose *S* is an input sentence, *(A, B)* is the selected example. The process of building links from *S's* fragments to *B's* fragments consists of two steps.

(1) Build links from *S's* fragments to *A's* fragments using a revised edit distance algorithm as shown in figure 2. Its result is denoted as *LinkSet(S→A)*.
(2) Build links from *S's* fragments to *B's* fragments (denoted as *LinkSet(S→B)*) according to following rules. (a) For a link in *LinkSet(S→A)*, if neither its source fragment nor its target fragment is null, replace its target fragment with this target fragment's corresponding aligned fragment in *B*, and add this new link to *LinkSet(S→B)*. (b) For a link in *LinkSet(S→A)* whose target fragment is null, add it to *LinkSet(S→B)* directly. (c) For those fragments in *B* that have not been linked, build links for each of them by assigning a null source fragment and a *D* link type to them respectively, and add these links to *LinkSet(S→B)*. (d) Reorder the items of *LinkSet(S→B)* in their target fragments' order in sentence *B*.

In the revised edit distance algorithm, it takes fragments as comparison units, and its two input sentences *S* and *A* are segmented into fragments by two segmentation tools[1] before they are inputted. This is a little different from Brown (1996) who took a full segmentation strategy for the input sentence.

```
m=length(S1), n=length(S2)
d[0][0] =0; tags[0][0] = 0;
for i=1 to m
  d[i][0]=q+d[i-1][0]; tags[i][0]='D'
for j=1 to n
  d[0][j]=r+d[0][j-1]; tags[0][j]='I'
for i=1 to m
 for j=1 to n
   p = computeCost(S1[i-1],S2[j-1]);
   a = d[i-1][j-1] + p;
   b=d[i-1][j] + q;
   c=d[i][j-1] + r;
   d[i][j] = min(a,b,c);
   if(min==a and p==0)
      tags[i][j] = 'N';
   else if (min==a)
      tags[i][j] = 'R';
   else if (min==b)
      tags[i][j] = 'D';
   else if (min==c)
      tags[i][j] = 'I';
return tags
```

Figure 2. Revised Edit Distance Algorithm

In figure 2, ***computeCost*** is a function to compute two fragments' linking cost based on their lexical forms and their head words' POSs. Its possible value belongs to the range [0, 1] and is manually assigned according to human's experiences. If two fragments' lexical forms are the same and their head words' POSs are the same too, this cost is zero; if two fragments' lexical forms are the same but their head words' POSs are different, this cost is 0.2; otherwise, this value is assigned by human's experiences according to the two fragments' head words' POSs as shown in table 1.

Table 1. Linking Cost for Two Fragments

$PosPair(c_i, c_j)$	w_i
(noun, noun)	0.5
(noun, auxiliary)	0.8
(noun, adjective)	0.85
…	…

[1] http://chasen.aist-nara.ac.jp/hiki/ChaSen/ for Japanese
http://www.nlplab.com/chinese/source.htm for Chinese

In figure 2, q, r are constants. It is required that $q + r \geq p$ and $q,r \in (0,1]$, here we set $q = r = 1$. The returned ***tags*** is *LinkSet(S→A)*.

After step 1, we can build links from sentence S to sentence B according to the rules described in step 2, and an example of this process is shown in figure 3.

Suppose *S* is "他很爱他的妻子*(He loves his wife very much)*". The selected example *(A,B)* is "(他爱他的妈妈*(He loves his mother)*, 彼は、彼の母を愛しています*(He loves his mother)*)".

Firstly, *LinkSet(S→A)* is built using the algorithm shown in figure 2. It is: (他*(he)*,他,*N*), (很*(very much)*,null,*I*), (爱*(loves)*,爱,*N*), (他的*(his)*,他的, *N*), (妻子*(wife)*,妈妈*(mother)*,*R*).

Secondly, *LinkSet(S→B)* is built as follows. We know that in *(A,B)*, "他"aligns to "彼*(he)*", "爱" aligns to "愛しています*(loves)*" , "他的" aligns to "彼の*(his)*", and "妈妈" aligns to "母*(mother)*", according to rule (a), we replace these target fragments in *LinkSet(S→A)* with their corresponding aligned fragments in *B* and add them to *LinkSet(S→B)*, and *LinkSet(S→B)* is changed to: (他*(he)*, 彼*(he)*,*N*), (爱*(loves)*, 愛しています*(loves)*,*N*), (他的*(his)*,彼の*(his)*,*N*), (妻子*(wife)*,母*(mother)*,*R*). For the link (很*(very much)*,null,*I*), according to rule (b), we add it to *LinkSet(S→B)* directly. Besides, there are some fragments in *B* that haven't been linked, according to rule (c), we build links for each of them by assigning them a null source fragment and a link type *D*, and add these new links in *LinkSet(S→B)* , and *LinkSet(S→B)* is changed to: (他*(he)*,彼*(he)*,*N*), (爱*(loves)*,愛しています*(loves)*,*N*), (他的*(his)*, 彼の*(his)*, *N*), (妻子(wife), 母*(mother)*,*R*), (很*(very much)*,null,*I*), (null,は*(ha)*,*D*), (null,を*(wo)*,*D*). At last, according to rule (d), we reorder the items in *LinkSet(S→B)*, and the final *LinkSet(S→B)* is: (他*(he)*, 彼*(he)*,*N*), (null, は*(ha)*,*D*), (很*(very much)* ,null,*I*), (他的*(his)*, 彼の*(his)*, *N*), (妻子(wife), 母*(mother)*,*R*), (null,を*(wo)*,*D*), (爱*(loves)*,愛しています*(loves)*,*N*).

Figure 3. An Example of Building Links

3.2 States Assignment

3.2.1 States for Non-*I* Type's Links

If a link's type is not *I*, that is to say it is one of the types {*R, D, N*}, the state assignment is easy. If its link type is *R*, a state named *S_R* is assigned;

if its link type is *D*, a state named *S_D* is assigned; if its link type is *N*, a state named *S_N* is assigned.

3.2.2 States for *I*-Type's Link

For an *I*-type's link, it indicates a possible generation operation is inserting. Different from other link types, there are two challenges for it: one is how to select a proper inserting position; the other is how to make the whole sentence fluent when finishing this inserting operation. In response to these two problems, we use current *I*-type link's pre- and post- links' link shapes to define current *I*-type link's state.

Suppose an *I*-type's link in *LinkSet(S→B)* is (*i*, null,*I*), *i*+1 and *i*-1 are the post- and pre- fragments of this link's source fragment. *m* and *n* are some fragments in sentence *B*. It is the same that we use $m \pm 1$ and $n \pm 1$ to denote the post- and pre- fragments of m and n respectively.

According to the link shapes of the links that take *i*+1 and *i*-1 as their source fragments, there are twelve basic link shapes shown in figure 4 and three extended link shapes shown in figure 5. We map each of these link shapes to an *I*-type link's state. Thus there are twelve basic states and three extended states for *I*-type's links.

In figure 4 and figure 5, a dot rectangle denotes a true link in *LinkSet(S→B)*, and a bold rectangle denotes this link's generation path when taking into account *LinkSet(S→A)*.

A brief explanation to these states is as follows. For example, state 6 in figure 4 means *S's* fragment *i*-1 links to *B's* fragment *m* and *S's* fragment *i+1* links to nothing in *B*. The appearance reason for this null target fragment is that in sentence pair *(S,A)*, fragment *i+1* links to fragment b_j, but in sentence pair *(A,B)*, b_j aligns to null, thus *i+1* links to null according to the second step when building *LinkSet(S→B)*. Due to the same or similar reason, state 7, 8, 10, 12,13,14,15 also have null target fragments in their links. We distinguish these link shapes because they will be treated differently. State 9 indicates that *i* is the first fragment in sentence *S*. State 11 indicates that *i* is the last fragment in sentence *S*.

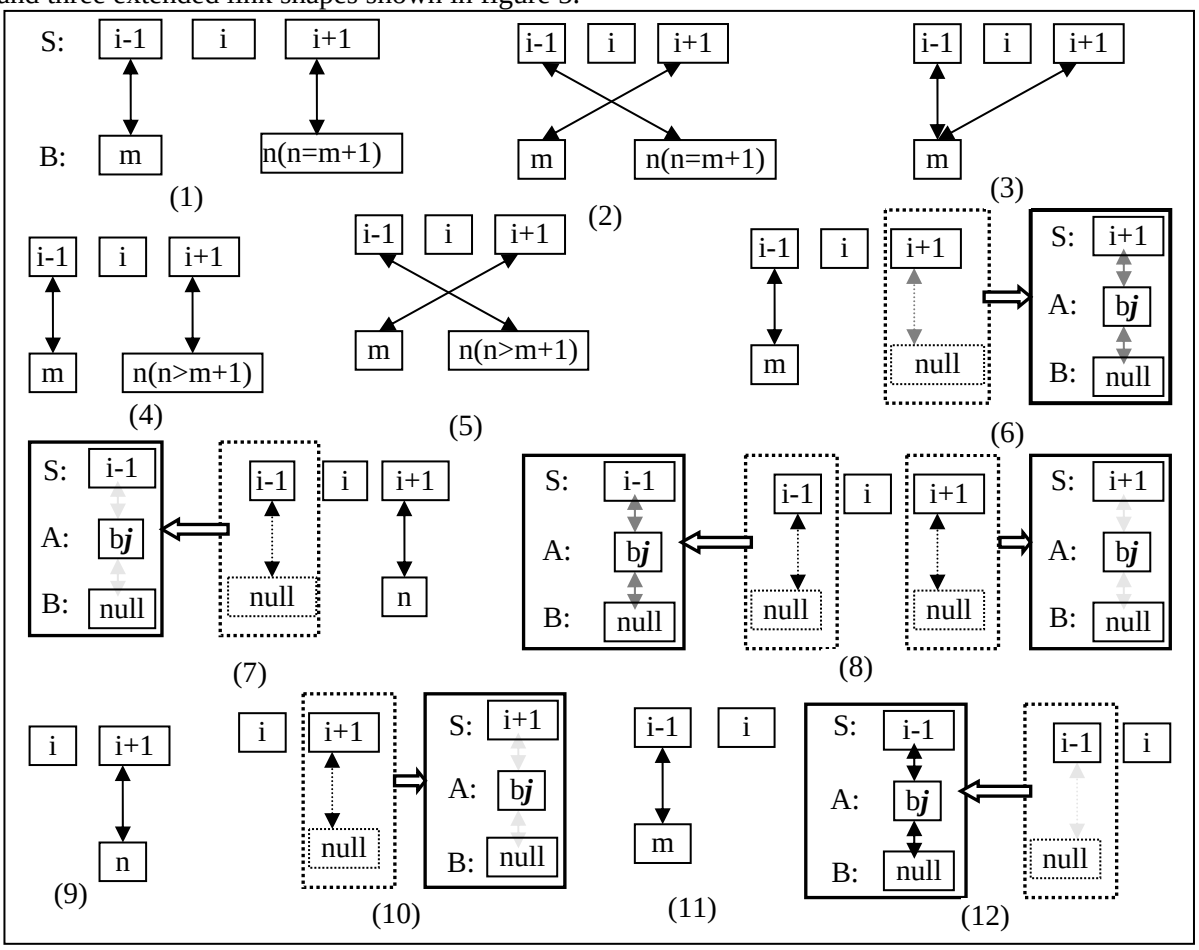

Figure 4. Basic States for *I*-type's Link

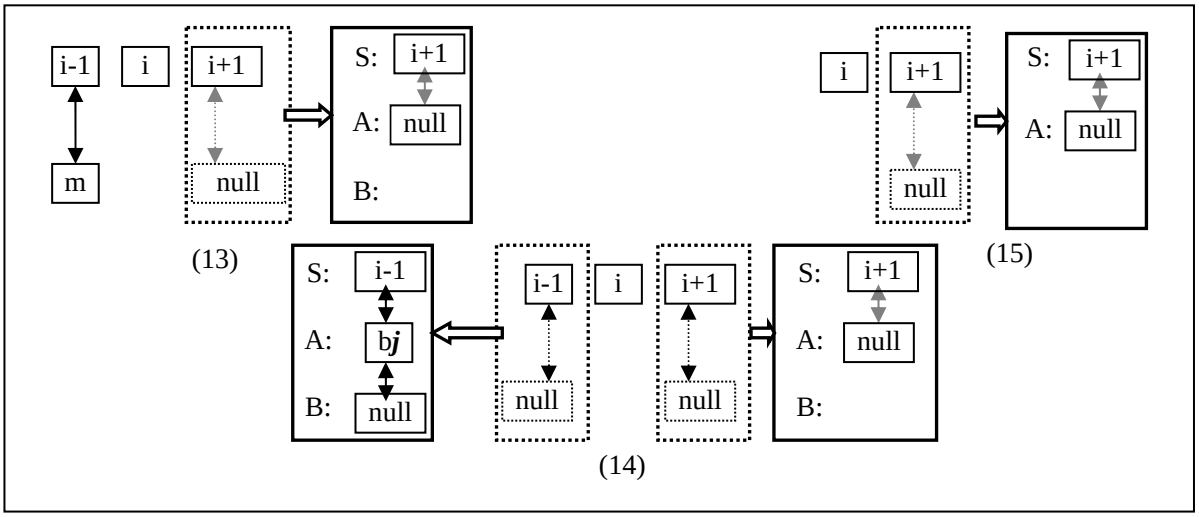

Figure 5. Extended States for *I*-type's Link

In practice, we will meet the extended states in figure 5, but they can be converted into basic states in some way. These conversion rules are as follows. For state 13, move rightward until find a non-*I* type's link, if this link's target fragment is null, convert it to state 6; otherwise, convert it to a state among state 1 to state 5 according to the link shapes of fragment *i*-1's link and the new found link; if can't find a non-*I* type's link in current link's right side, convert it to state 11. For state 14, move rightward until find a non-*I* type's link, if this link's target fragment is null, convert it to state 8, otherwise, convert it to state 7; if can't find a non-*I* type's link in current link's right side, convert it to state 12. For state 15, move rightward until find a non-*I* type's link, if this link's target fragment is null, convert it to state 10, otherwise, convert it to state 9; if can't find a non-*I* type's link in current link's right side, move leftward until find a non-*I* type's link (this link will be found always) and convert it to state 11.

For all these conversions, the final new state's *I*-type link takes all the passed fragments in *S* during rightward movement as its new source fragment.

By conversion, every *I*-type's link can be mapped to a basic state in figure 4, and we can consider basic states only in the following description.

3.3 Translation Generation

In this process, an automaton is constructed to generate the input sentence's translation. For different state, there is different generation operation corresponds to it.

3.3.1 Generation Operations for Non-*I* Type Links' States

If a link's type is not *I*, we take an easy generation strategy according to its state. If a link's state is *S_R*, replace this link's target fragment with its source fragment's translation, and denote this operation as ***O(R)***; if a link's state is *S_D*, delete this link's target fragment, and denotes this operation as ***O(D)***; if a link's state is *S_N*, remain this link's target fragment unchanged, and denote this operation as ***O(N)***. Here a link's source fragment's translation is generated by looking up a dictionary.

3.3.2 Generation Operations for *I*-Type Links' States

If a link's type is *I* (suppose its source fragment is *i*), we take its source fragment's pre- and post-fragments into account and judge: whether the fragment combinations *(i-1,i,i+1)*, *(i-1,i)* and *(i,i+1)* are chunks. If they are chunks, look up their corresponding translations in dictionary, otherwise, look up *i's* translation in dictionary (we assume its translation can be found always). Here a chunk is defined as a translation unit and a simple dictionary-based method is used for chunk recognition: as long as a fragment can be found in dictionary, it is regarded as a chunk. According to current *I*-type link's state and the recognized chunk information, we choose one of these chunks as current *I*-type link's new source fragment for later processing, and define 10 possible generation operations as follows.

- ***O(0):*** Delete the links that take *B's* fragments among *m+1* to *n* as their target

fragments. And for the link that takes *B's* fragment *m* as target fragment, replace *m* with the translation of current *I*-type link's new source fragment.

- ***O(1***): For the link that takes *B's* fragment *m* as target fragment, replace *m* with the translation of current *I*-type link's new source fragment.
- ***O(2):*** For the link that takes *B's* fragment *n* as target fragment, replace *n* with the translation of current *I*-type link's new source fragment.
- ***O(3):*** For the link that takes *B's* fragment *m* as target fragment, add the translation of current *I*-type link's new source fragment to the end of *m*.
- ***O(4):*** For the link that takes *B's* fragment *n* as target fragment, add the translation of current *I*-type link's new source fragment to the end of *n*.
- ***O(5):*** For the link that takes *B's* fragment *m* as target fragment, replace *m* with the translation of current *I*-type link's new source fragment. And delete the link that takes *B's* fragment *n* as target fragment.
- ***O(6):*** For the link that takes *B's* fragment *n* as target fragment, replace *n* with the translation of current *I*-type link's new source fragment. And delete the link that takes *B's* fragment *m* as target fragment.
- ***O(7):*** For the link that takes *B's* fragment *m* as target fragment, add the translation of current *I*-type link's new source fragment before *m*.
- ***O(8):*** For the link that takes *B's* fragment *n* as target fragment, add the translation of current *I*-type link's new source fragment before *n*.
- ***O(9):*** Do not modify any link's target fragment.

Here *m* and *n* are sentence *B's* fragments, and they also correspond to the target fragments of the links shown in figure 4.

During the generation, which operation should be chosen depends on current *I*-type link's state and the result of chunk recognition. The choice strategy will be described subsequently.

3.3.3 Finite Automaton State Transfer Based Generation

Based on *LinkSet(S→B)* and the assigned states, we construct an automaton that has a similar form as shown in figure 6. This automaton takes *LinkSet(S→B)* and the assigned states as input, executes generation operations according to these states and outputs *LinkSet(S→B)'s* final modified target fragment sequence as the input sentence's translation result.

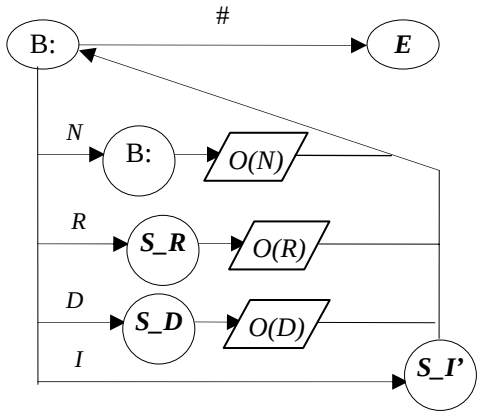

Figure 6. Finite Automation State Transfer Based Generation

In figure 6, ***B*** is a start state, ***E*** is an end state, {*I, R, D, N*} are link types, {*O(N), O(D), O(R)*} in parallelogram are the operations defined in section 3.2.1; and # is a fictitious symbol that indicates the end of the automaton's input. {*S_R, S_D, S_N*} are states correspond to non-*I* type's links. And *S_I'* is a state set that corresponds to *I*-type's links. When the state transfers to *S_I'*, the corresponding operations are shown in figure 6. In this figure, numbers from *1* to *12* in ellipse circles correspond to the states defined in figure 4. *O(i)* in parallelogram corresponds to the operations defined in section 3.3.2; *O'* in the operation of state 3 means the automaton generates the fragment combination *(i-1,i,i+1)'s* translation by simply joining their single fragment's translations together. d_1 means the semantic distance from fragment *i* to fragment *i-1*, and d_2 means the semantic distance from fragment *i* to fragment *i+1*, and they are computed as shown in formula 2.

$$dist(f_1, f_2) = \sum_{c_i \in f_1} \sum_{c_j \in f_2} w_k(PosPair(c_i, c_j)) \quad (2)$$

In formula 2, f_1 and f_2 are fragments, c_i and c_j are words in them, w_k is a weight function whose value is determined by the POSs of words c_i and c_j, and its value assignment strategy can be referred to table 1. When current *I*-type link's pre- and post- links' target fragments span several fragments, this formula is used to identify a proper inserting position for the translation of current *I*-type link's source fragment. The larger this distance, the less possibility its two fragments' translations are close.

Figure 7 shows the operation strategy for different states of the *I*-type's links. Here we take state 1 as example and give some explanations for these operations in figure 6. For state 1, if the fragment combination *(i-1,i,i+1)* is a chunk, from the link shape of state 1 in figure 4 we can see, there is a strong hint that the original target fragments of the two links that take fragments *i-1* and *i+1* as their source fragments respectively should be replaced by this new chunk's translation, and this just corresponds to the first operation defined in section 3.3.2. Otherwise, if *(i-1,i)* is a chunk, there is a strong hint that the original target fragment of the link that takes *i-1* as its source fragment should be replaced by this new chunk's translation; and other cases are similar to these explanations.

The main idea for the operation strategies in figure 7 is trying to enlarge the source fragment for an *I*-type's link, and using its contextual links' link shapes to find a proper inserting position for the translation of its new source fragment.

To demonstrate this generation process, we continue the example introduced in section 3.1.

After building links described in section 3.1 *LinkSet(S→B)* is: (他*(he)*, 彼*(he)*,*N*), (null, は*(ha)*,*D*), (很*(very much)*,null,*I*), (他的*(his)*, 彼の*(his)*,*N*), (妻子(wife),母*(mother)*, *R*), (null, を*(wo)*,*D*) (爱*(loves)*,愛しています*(loves)*,*N*).

Its corresponding state sequence is: *S_N, S_D, S_I_4(*the forth state in figure 4*),S_N, S_R, S_D, S_N*.

During the process of generation, the constructed automaton takes *LinkSet(S→B)* and the corresponding state sequence for the links in *LinkSet(S→B)* as inputs, and analyzes these inputs one by one. This process is shown in figure 8 which give an example of the translation generation process.

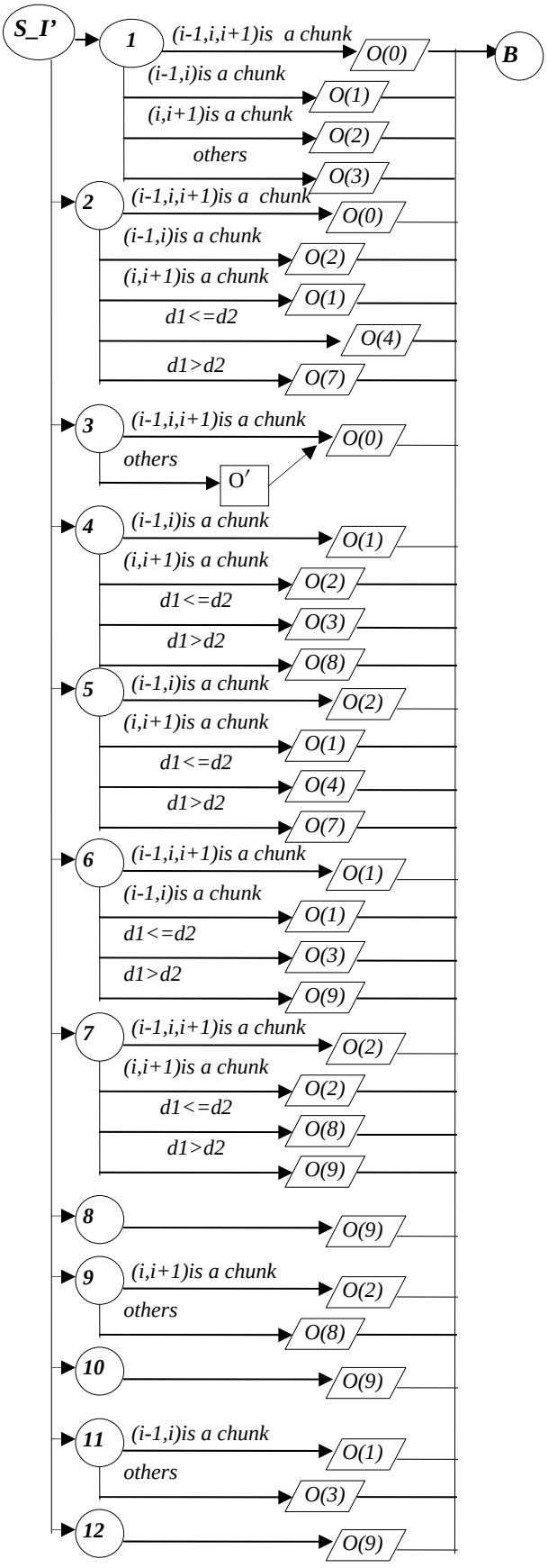

Figure 7. State Transfer for *I*-Type's Links

For the link (他*(he)*,彼*(he)*,*N*), its state is *S_N*. According to figure 6, the automaton executes operation *O(N)* and does not modify this link's target fragment.

For the link (null,は*(ha)*,*D*), its state is *S_D*. According to figure 6, the automaton executes operation *O(D)* and deletes this link's target fragment.

For the link (很*(very much)*,null,*I*), its state is *S_I_4*. If the fragment combination *(i-1,i)* "他很*(he…very much)*" is a chunk and the corresponding translation is "彼は、とても *(he…very much)* ", according to figure 6, the automaton executes operation *O(1)*. It first takes this recognized chunk as current link's new source fragment. Then it selects the link whose target fragment is "彼*(he)*", and this link is (他*(he)*,彼*(he)*,*N*). Thirdly, it replaces the selected link's target fragment with the translation of current *I*-type link's new source fragment. At last the selected link is changed to (他*(he)*, 彼は、とても*(he…very much)*, *N*).

For the link (他的*(his)*,彼の母*(his)*,*N*), its state is *S_N*. According to figure 6, the automaton executes operation *O(N)* and does not modify this link's target fragment.

For the link (妻子*(wife)*,母*(mother)*,*R*), its state is *S_R*. According to figure 6, the automaton executes operation *O(R)* and replaces this link's target fragment with its source fragment's translation. Finally current link is changed to (妻子*(wife)*,妻*(wife)*,*R*).

For the link (null,を*(wo)*,*D*), its state is *S_D*. According to figure 6, the automaton executes operation *O(D)* and deletes this link's target fragment.

For the link (愛*(loves)*,愛しています*(loves)*,*N*), its state is *S_N*. According to figure 6, the automaton executes operation *O(N)* and does not modify this link's target fragment.

At last, the automaton ends the state transfer process and outputs *LinkSet(S→B)'s* modified target fragment sequence "彼は、とても彼の妻愛しています*(he loves his wife very much)*" and takes it as the input sentence's translation.

Figure 8. An Example of Generation

4 Experiments

We developed a Chinese-Japanese bidirectional EBMT system to evaluate the proposed method in term of translation quality, and BLEU value and NIST score are used for evaluation. The evaluation tool is the NIST MT Evaluation Toolkit[2].

4.1 System Resources

Bilingual Corpus We collect 10083 Chinese-Japanese bilingual sentences from Internet in Olympic domain as examples. The average length of the Chinese sentences is 12.8 characters while the average length of the Japanese sentences is 25.6 characters. All the examples are stored in their lexical form along with their fragments alignment information. We used an in-house tool for fragment alignment and revised this result by some experienced experts.

Bilingual Dictionary A bilingual dictionary is used to translate the input fragment and to judge whether an input fragment is a chunk.

This bilingual dictionary contains not only the general word items, but also some bilingual chunks collected from our corpus by an in-house rule-based chunk parser. All together there are about 150,000 word items and about 71,000 chunk items in this bilingual dictionary.

Language Model During the process of *R*-type and *I*-type links' generations, if a fragment has several translations, a language model is used for its translation choice (Feiliang Ren and Tianshun Yao, 2006). Its work principle is to make the whole sentence fluent most after fragments translation choices. For example, if during the process of translation generation, we need to insert a fragment's translation into the target part of the selected translation example, and if there are several different translations for this fragment in dictionary, which translation should be chosen? Our method is to choose the one that can make the final sentence fluent most after choices. And use language model to measure the fluency of a sentence.

We collected an approximate 1,400,000 words' Japanese monolingual corpus and a similar size's Chinese monolingual corpus from Internet, and trained a standard trigram Japanese language model for Chinese-to-Japanese EBMT system and a standard trigram Chinese language model for Japanese-to-Chinese EBMT system respectively.

Test Corpus We collect another 100 bilingual sentences in Olympic domain from Internet as

[2] ttp://www.nist.gov/speech/tests/mt/resources/scoring.htm

test corpus. But it is required that for every sentence *S* in test corpus, there must be at least one example *(A, B)* that satisfies $Sim(S, A) \geq 0.4$. This is because the characteristic of EBMT is translated by analogy. If there weren't any proper examples for the input sentence, the advantage of EBMT would vanish. When this happened, system would have to perform translation in a different manner. This is not what we hope. This threshold condition can guarantee that system performs translation in an EBMT manner and thus we can focus on the generation method proposed.

4.2 Experimental Results

We take system's matching module as a baseline system. In fact it is a TM (translation memory) system, and its performance is the lowest limit of our translation system's performance.

In the evaluation, we set N at 4 when computing BLEU value and NIST score. Experimental results for Chinese-to-Japanese EBMT system and Japanese-to-Chinese EBMT system are shown in table 2 and table 3 respectively.

Table 2. Experimental Results for Chinese-to-Japanese EBMT System

Method	NIST	BLEU
Baseline	4.8321	0.4913
Our System	***5.9729***	***0.7705***

Table 3. Experimental Results for Japanese-to-Chinese EBMT System

Method	NIST	BLEU
Baseline	4.1275	0.4076
Our System	***5.0976***	***0.5908***

From table 2 and table 3, it can be seen that our system achieves excellent translation performances in both Chinese-to-Japanese translation system and Japanese-to-Chinese translation system. These results are unexpected and encouraging. We think the following reasons lead to these good results. First, we set a threshold in matching module. This guarantees that even under the worst condition, our system's performance is still at a relative high level. Second, the alignment results for the fragments of the examples stored in corpus are revised by experienced experts. It makes the alignment precision be very high. And this is very helpful when building links before generation. Third, we generate the translation by modifying the target sentence of the selected example, this makes us use the existed target sentence's structure information as much as possible, and it is useful for generating translation that conforms to the grammar and the semantic rules well. Forth, the most important point is that we view the generation as a process of finite automata state transfer, search out the most useful information for the input fragments by building fragments' links from the input sentence to the target sentence of the selected example, and take different generation strategies for different kinds of states.

We also notice that the performance of Chinese-to-Japanese translation system is better than the performance of Japanese-to-Chinese translation system. This is because that generally a Japanese sentence has a more complicated structure than a Chinese sentence. This will lead to poorer result when building fragments' links from sentence *S* to sentence *A*, thus the fragments' links from *S* to *B* are worse accordingly. So the final translation result will be worse because the proposed method is affected by the link result heavily. More work should be done to improve the algorithm that builds links from *S's* fragments to *A's* fragments.

Besides, there are still some translation results that are not as good as expected. For example, in the Chinese-to-Japanese translation system, some auxiliary particles were wrongly deleted, which made several translation results were somewhat odd when checked by a Japanese native speaker. This is caused by the simple deleting strategy in our generation process for those *D*-type's links. We think that operation strategy for these *D*-type's links needs further improvement.

5 Conclusions and Future Work

This paper proposes an EBMT method based on finite automata state transfer generation. During the translation process, first a bilingual sentence pair is selected as example whose source sentence is most similar to the input sentence; then the target sentence of this example is used to generate final translation result in a finite automata state transfer manner. During the generation process, firstly we build links from the fragments in the input sentence to the fragments in the target sentence of the selected example. Then we assign states for each of these links. Finally, we construct a finite automaton with these states and generate a translation result in a finite automata state transfer manner. Our method hasn't any special requirement for corpus's domain. It can

be easily replicated, and does not need some complicated parsers either. As long as you have a bilingual corpus and a fragment alignment technology (even it is a simple dictionary-based method), you can replicate our work. Therefore, we think it is a good baseline method for machine translation.

From the generation process and experimental results we can see that there are some factors that affect our translation system's performance heavily, such as the algorithm used to build links, the similarity algorithm for matching module, the fragment alignment technology, and the chunk recognition method and the translation generation technology for the recognized chunks, and so on. In future work, we will investigate improving the performances of these factors.

Acknowledgments

This work was supported in part by the National Natural Science Foundation of China under Grant No.60473140; the National 863 High-tech Project No.2006AA01Z154; the Program for New Century Excellent Talents in University No.NCET-05-0287; and the National 985 Project No.985-2-DB-C03.

References

Eiji Aramaki, Sadao Kurohashi. 2003. Word Selection for EBMT based on Monolingual Similarity and Translation Confidence. In Proceedings of the HLT-NAACL 2003 Workshop on Building and Using Parallel Texts, pp 57-64

Eiji Aramaki, Sadao Kurohashi. 2004. Example-Based Machine Translation Using Structural Examples. International Workshop on Spoken Language Translation (IWSLT), pp. 91-94.

Eric Sven Ristad, Peter N.Yianilos. 1998. Learning string-edit distance. IEEE Transactions on Pattern Analysis and Machine Intelligence. 20(5):522-532

Feiliang Ren, Tianshun Yao. 2006. Make Word Sense Disambiguation in EBMT Practical. The 20th Pacific Asia Conference on Language, Information and Computation. pp414-417

Harold Somers.1999. Review Article: Example-based Machine Translation. Machine Translation 14, pp.113-157

Kenji Imamura, Hideo Okuma, Taro Watanabe and Eiichiro Sumita. 2004. Example-Based Machine Translation Based on Syntactic Transducer with Statistical Models. In Proc. of the 20th International Conference on Computational Linguistics (COLING-2004), pp. 99-105.

Kevin Knight, Yaser Al-Onaizan, 1998. Translation with Finite-State Devices, Proceedings of the AMTA Conference, Langhorne, PA, USA. pp.421-437.

Liu Zhanyi, Wang Haifeng, Wu Hua. 2005. Example-based machine translation based on TSC and statistical generation. MT Summit X, pp.25-32

LV Xue-qiang, Ren Feiliang, 2003. Sentence Similarity Model and the Most Similar Sentence Search Algorithm. Journal of Northeastern University (Natural Science). Pp531-534

Ralf D. Brown, 1996. Example-Based Machine Translation in the Pangloss System, Procddeings of the 16th International Conference on Computational Linguistics(COLING-96), pp.169-174

Robert A.Wagner, Michael J.Fischer. 1974. The String-to-String Correction Problem, Journal of the ACM(JACM), v.21, pp.168-173

Shankar Kumar, William Byrne. 2003. A Weighted Finite state automata transfer Implementation of the Alignment Template Model for Statistical Machine Translation. Proceedings of the Conference on Human Language Technology. Edmonton, Canada. Pp.142-149.

Shankar Kumar, William Byrne, 2004. A Wighted Finite state automata transfer Example Model for Statistical Machine Translation. Natural Language Engineering

Srinivas Bangalore, Giuseppe Riccardi. 2001. A finite state approach to machine translation. Proceedings of the 2nd meeting of North American Chapter of the Association for Computational Linguistics. Pittsburgh, PA, USA.

Taro Watanabe, Ei ichiro Sumita. 2003. Example-based decoding for statistical machine translation. In Proceedings of MT Summit IX, pp.410-417

T.H.Cormen, C.E.Leiserson, R.L. Rivest and C.Stein. 2001. Introduction to Algorithms (the second edition). The MIT Press

Yasuhiro Akiba, Taro Watanabe and Eiichiro Sumita. 2002. Using Language and Translation Models to Select the Best among Outputs from Multiple MT systems. In Proceedings. of the 19th International Conference on Computational Linguistics (COLING-2002), pp. 8-14.