Graph-based Retrieval for Example-based Machine Translation Using Edit-distance

Takao Doi, Hirofumi Yamamoto and Eiichiro Sumita
ATR Spoken Language Communication Research Laboratories
2-2-2 Hikaridai, Kansai Science City, Kyoto, 619-0288 Japan
{takao.doi, hirofumi.yamamoto, eiichiro.sumita}@atr.jp

Abstract

An EBMT system whose translation example unit is a sentence, can produce an accurate and natural translation if translation examples similar enough to an input sentence are retrieved. Such a system, however, suffers from the problem of narrow coverage. To reduce this disadvantage, a large-scale parallel corpus is required, which calls for an efficient retrieval method. The authors propose a method for a sentence-wise EBMT system to efficiently retrieve the most similar sentences using the measure of edit-distance without omissions. The proposed method uses search space division, word graphs and an A* search algorithm. The performance of the EBMT system implemented with the method was evaluated through Japanese-to-English translation experiments using a bilingual corpus comprising hundreds of thousands of sentences from a travel conversation domain. The EBMT system achieved a high-quality translation ability by using a large corpus, and also achieved efficient processing by using the proposed retrieval method.

1 Introduction

An Example-Based Machine Translation (EBMT) system retrieves the translation examples that are most similar to an input expression and adjusts the examples to obtain the translation. The translation example unit is usually a phrase, and the translations of phrases are combined into a translation of the input sentence. Although a phrase looks like a suitable translation example unit because of its generality for covering various sentences, there is a risk of mixing errors or producing unnatural translations while combining phrases into a sentence (Somers, 2003). Such risk, however, can be reduced if the translation example unit is a sentence. A translation example similar enough to an input sentence as a whole results in an accurate and natural translation of the input sentence. Of course, an EBMT system whose translation example unit is a sentence (hereafter, we call this a sentence-wise EBMT system), suffers from the problem of narrow coverage because a sentence is a longer unit and its generality is lower. To achieve sufficiently broad translation coverage by using sentence-wise EBMT, we must prepare a large-scale parallel corpus and, therefore, an efficient method is needed to retrieve translation examples from a large-scale corpus.

In this paper, we propose a retrieval method for a sentence-wise EBMT system, whose measure of similarity is edit-distance. An efficient retrieval method for EBMT has much in common with Translation Memory (TM). Research efforts on TM (Sato, 1992; Cranias et al., 1997; Planas and Furuse, 1999; Baldwin and Tanaka, 2001) focus on filtering algorithms for sentence sets and/or matching algorithms between two sentences. The methods adopting filtering algorithms, first, filter sentences out by using, for example, a clustering technique that applies word vectors. Then, for each of the sentences left as candidates, they repeat the matching procedure between the two sentences, a candidate and the input. Unfortunately, these methods sometimes omit the most similar sentences in the filtering process. On the other hand, this paper proposes an efficient retrieval method without omissions, as a solution for the problem of searching for the sentences with the least edit-distance among a corpus. This method does not repeat the matching procedure between two sentences, but proceeds to match the input sentence and sentences in a corpus concurrently, where the sentences are expressed as a graph.

The following sections give an overview of the target EBMT system, an overview of the proposed retrieval method, a description of the search algorithm of the method, and a performance evaluation.

2 Target EBMT System

2.1 Overview

(Sumita, 2003) proposed the Dp-match Driven transDucer (D^3), a sentence-wise EBMT method, in which translation examples are sentence pairs of source and target languages. When translating an input sentence, the system retrieves the translation examples whose source sentences are the most similar to the input sentence using the measure of edit-distance. Translation patterns are then dynamically generated with consideration of differences between the input sentence and the translation examples. D^3 keeps and retrieves translation examples that are not abstracted more than the word sequences given in a corpus. Furthermore, the changes in the target sentences of translation examples are kept as small as possible while translations are generated. Therefore, natural translations occur if there are examples similar enough to given input sentences.

2.2 Example Retrieval

Among the processing phases of D^3, the example retrieval phase tends to take the greatest portion of the translation processing time. The function of example retrieval is to find the examples with the minimum distance in the bilingual corpus. This distance is measured between word sequences of the input sentence and the source sentence of an example.

The distance between word sequences is defined as *dist* in Eq. (1), which is the edit-distance including a semantic factor. In this equation, L_{input} and $L_{example}$ respectively indicate the number of words in the input sentence and that in the source sentence of the example, while I and D denote the numbers of insertions and deletions, respectively. Substitution is considered as the semantic distance between two substituted words, described as $SEMDIST$. Substitutions are permitted only between the content words of a common part of speech. Following this equation, *dist* is the total value of insertions, deletions and substitutions normalized by the lengths of the word sequences. $SEMDIST$ is defined using a thesaurus and ranges from 0 to 1. Furthermore, $SEMDIST$ is the division of K (the level of the least common abstraction of two words in the thesaurus) by N (the height of the thesaurus) according to Eq. (2) (Sumita and Iida, 1991).

$$dist = \frac{I + D + 2\sum SEMDIST}{L_{input} + L_{example}} \qquad (1)$$

$$SEMDIST = \frac{K}{N} \qquad (2)$$

If the minimum distance is not small enough, the examples are not useful for translation. Therefore, we use a threshold for the distance. If there are no examples within the given threshold, the retrieval, and furthermore, the whole translation process fails with no output.

3 Proposed Retrieval Method

The function of our target retrieval process is to retrieve every sentence whose edit-distance *dist* against a given input sentence is the least and falls within a given threshold, from among the candidate sentences, which are all of the source sentences in translation examples. This distance is defined by the relation of two sentences and can be calculated using DP-matching (Cormen et al., 1989) between the two sentences. Therefore, the candidate sentences with the least distance can be found by repeating DP-matching between a candidate and the input. However, because the procedure of such a naive algorithm takes time proportional to the number of translation examples, it is difficult to implement real-time processing for translation using a large-scale corpus on ordinary computers. Consequently, we propose an efficient retrieval method using the classification of candidates, word graphs and an A* search algorithm. This method does not make omissions; that is, in the definition of the distance *dist* and a given threshold, the retrieval result of this method is the same set of sentences as the case of using DP-matching sequentially.

3.1 Candidate Set Classification

Candidate sentences are classified by the number of content words and the number of functional words. This makes it possible to filter candidates according to the numbers of content words and functional words in the input sentence and the distance threshold. That is, the least possible distance can be calculated on the assumption that all content words are the same as each other, and all functional words are also the same as each other. The classes of the least possible distance greater than the threshold are filtered out. The smaller the least possible distance of the class, the sooner the search process is applied to the class. If a candidate of distance smaller than the threshold is found in a class, the threshold is updated with the distance of that candidate. The smaller threshold can filter out more classes. Furthermore, the search algorithm in a class, which is described in Sect. 4, utilizes the precondition that all sentences in a class have the same number of content words and the same number of functional words, and therefore, the same number of words.

3.2 Word Graph

For each group classified by the numbers of content words and functional words, a word graph is composed of all candidate sentences in the class. Figure 1 illustrates an example of a word graph. The word graph is a directed graph and has a start node and a goal node. Each possible path from the start node to the goal node corresponds to a candidate sentence. Common word sequences in multiple sentences share the same edges. The word graph is compressed so that the number of nodes is the minimum by the method of converting finite state automata (Brzozowski, 1962). By using the word graph, the search process scans all the sentences in a class concurrently.

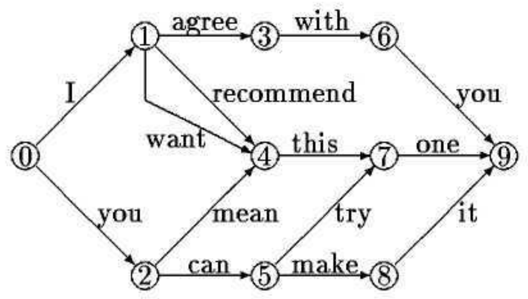

Figure 1: Example of Word Graph

3.3 A* Search Algorithm

The result of matching between two word sequences is represented as a sequence of substitutions, insertions and deletions. We call the result a matching-sequence. The search process in a class is to search for the matching-sequences of the least distance among all possible matching-sequences between the input sentence and each sentence of the class. We use an A* search algorithm (Nilsson, 1971) to solve the search problem.

Generally in an A* search algorithm, the state of the least estimated cost is selected and extended into successor states. In our search problem, the state means an incomplete matching-sequence between the input sentence and a path from the start node to the goal node in a word graph.

4 Search

In this section, we focus on the search process using a word graph. A word graph consists of nodes and edges, and it has a start node and a goal node. An edge consists of a word as label, a source node and a destination node.

4.1 State Space Representation

We represent the search problem using the state structure, operators and the definitions of initial state and goal state.

4.1.1 State

A state has four attributes: paths, node, input and trans, whose contents are as follows.

- `paths` : List of partial matching-sequences.
- `node` : Node in the word graph indicating that matching has proceeded until this node.
- `input` : Partial word sequence of the input sentence not used for matching yet.
- `trans` : Indicator of operators available to this state.

An exact match, a substitution, an insertion and a deletion in matching-sequences of the paths are represented as records including a label and a word or words: (E, *word*), (S, *graph word*, *input word*), (I, *input word*) and (D, *graph word*) respectively. The cost of a state is the cost of an arbitrary matching-sequence in the paths, where matching-sequences have the same cost. The cost of a matching-sequence is the sum of costs of the records in it. The costs are defined as 0 for an E record, 1 for an I record, and 1 for a D record. The cost for an S record is defined as twice the semantic distance between the words in the record, except that it is a small positive value when the distance is 0. [1] The small value gives us the minimum cost of an S record.

4.1.2 Operators

A state is expanded into a successor state by an operator. Five operators are defined as follows. While each T-operator and I-operator is applied to a state, each of the operators, E, S and D is applied to a combination of a state and

[1]This exception is set up in order to distinguish synonymy from identity.

an edge. In the following description, each operator is applied to a state s, if necessary, with an edge e, and s is expanded into a successor state s'. For each operator, we describe the condition where the operator can be applied, and the successor state to be generated as the result of the application.

- **T-operator** :
  - **condition** : s.trans is E-operator or S-operator.
  - **result** : s'.trans = selection from S-operator and NIL (described below) if s.trans is E-operator, NIL if s.trans is S-operator.
    Other attributes of s' are the same as those of s.
- **E-operator**— :
  - **condition** : s.trans is E-operator,
    and s.input is not empty,
    and e.source is s.node,
    and e.label and the head of s.input are identical.
  - **result** : s'.paths is generated by adding an E record to each element of s.paths.
    s'.node = e.destination.
    s'.input is generated by deleting the head of s.
    s'.trans = selection from E-operator, S-operator and NIL (described below.)
- **S-operator** :
  - **condition** : s.trans is S-operator,
    and s.input is not empty,
    and e.source is s.node,
    and e.label and the head of s.input are content words of the same part of speech but not identical,
    and the semantic distance between these two words is smaller than 1.
  - **result** : s'.paths is generated by adding an S record to each element of s.paths.
    s'.node = e.destination.
    s'.input is generated by deleting the head of s.
    s'.trans = selection from E-operator, S-operator and NIL.
- **I-operator** :
  - **condition** : s.trans is NIL,
    and s.input is not empty.
  - **result** : s'.paths is generated by adding an I record to each element of s.paths.
    s'.node = s.node
    s'.input is generated by deleting the head of s.
    s'.trans = selection from E-operator, S-operator and NIL.
- **D-operator** :
  - **condition** : s.trans is NIL,
    and s.paths includes such an matching-sequence whose last record is not an I record,
    and e.source is s.node.
  - **result** : s'.paths is generated from s.paths: first such matching-sequences, whose last records are I records, are deleted; second a D record is added to each matching-sequence left.
    s'.node = e.destination.
    s'.input = s.input.
    s'.trans = selection from E-operator, S-operator and NIL.

In the definitions above, the selection from S-operator and NIL means S-operator if there is a possibility that S-operator can be applied to s', and NIL otherwise. We judge that the possibility exists if the head of s'.input is a content word and there is an edge whose source is s'.node and whose label has the same part of speech but is not identical to the head of s'.input. The selection from E-operator, S-operator and NIL means E-operator if there is an edge whose source is s'.node and whose label is the head of s'.input, and otherwise, the same as the selection from S-operator and NIL. T-operator does not proceed to the actual matching process, but controls the application order of other operators through the trans attribute.

The second condition of D-operator prohibits a D record after an I record. That is, we make it a rule to put a D record before an I record in a sequence of I and D records in order to avoid the redundancy of the multiple appearance of substantially the same matching-sequences.

4.1.3 Initial State and Goal State

In the initial state, the paths attribute is a list of an empty matching-sequence, the node attribute is the start node, the input attribute is the whole word sequence of the input sentence, and the trans attribute is the E-operator. A goal state is such a state whose node attribute is the goal node and whose input attribute is empty.

4.2 Search Algorithm

From the state space formed using the definitions of the initial state, the operators and the goal states, we search for the goal states of the minimum cost. As an initial condition, an upper limit of cost is given, which is a given distance threshold multiplied by the sum of the lengths of the input sentence and a candidate sentence in the word graph.

4.2.1 Evaluation Function

We define the evaluation function f^* as follows.

$$f^*(s) = g(s) + h^*(s)$$

$g(s)$ is the cost from the initial state to the state s, which equals the cost of state and can be calculated from s.paths. If s is a goal state, $f^*(s) = g(s)$. $h^*(s)$ is the lower limit of cost that is taken from the state s to a goal state.

All sentences in a word graph have the same number of content words and the same number of functional words. Therefore, we can tell the numbers of content words in the input sentence, content words in the word graph, functional words in the input sentence, and functional words in the word graph, which are untreated in the state s. Here, these numbers are represented as C_{input}, C_{graph}, F_{input} and F_{graph} respectively. The lower limit of cost based on the numbers of untreated words is expressed as $h'(s)$ below.

$$h'(s) = |C_{input} - C_{graph}| + |F_{input} - F_{graph}|$$

Furthermore, on the assumption that one of the operators E, S, I and D is applied to the state s where the application of T-operator precedes if necessary, the lower limit of the cost from s to a goal state is expressed as $h''(s, o)$ where o indicates an operator. The value of $h''(s, o)$ is as follows for each operator of o.

- `E-operator` : $h'(s)$.
- `S-operator` : $h'(s)$ plus the minimum cost of an S record.
- `I-operator` : $|(C_{input} - 1) - C_{graph}| + |F_{input} - F_{graph}| + 1$ if the head of s.input is a content word, $|C_{input} - C_{graph}| + |(F_{input} - 1) - F_{graph}| + 1$ otherwise.
- `D-operator` :
  H_c+1 if there is no edge whose source is s.node and whose label is a functional word,
  H_f+1 if there is no edge whose source is s.node and whose label is a content word,
  1 plus the minimum value between H_c and H_f otherwise, where
  $H_c = |C_{input} - (C_{graph} - 1)| + |F_{input} - F_{graph}|$,
  $H_f = |C_{input} - C_{graph}| + |F_{input} - (F_{graph} - 1)|$.

By using these values, $h^*(s)$ is defined as: 1) $h''(s$,E-operator) if s.trans is E-operator; 2) the minimum value among $h''(s$,S-operator), $h''(s$,I-operator) and $h''(s$,D-operator) if s.trans is S-operator; and 3) the minimum value between $h''(s$,I-operator) and $h''(s$,D-operator) if s.trans is NIL.

4.2.2 Algorithm

The search algorithm is described below, where OPEN is a list of unexpanded states and CLOSED is a list of expanded states. The sameness of states in (5) means that two states are the same if they have the same value for each attribute except the paths.

1. set the value of cost upper limit and let OPEN be a list including the initial state alone.
2. terminate unless OPEN has a state of cost within the cost upper limit.
3. remove a state s of the least value of f^* from OPEN and put s into CLOSED.
4. if s is a goal state, keep s as a solution, change the value of cost upper limit with the cost of s and return to (2).
5. expand s into all of its successor states and for each successor state s', if $f^*(s')$ is within the cost upper limit, branch by the conditions:
   (a) if there is no same state as s' in either OPEN or CLOSED, put s' into OPEN;
   (b) if there is the same state as s' whose cost is larger than that of s' in OPEN or CLOSED, remove the same state and put s' into OPEN;

(c) if there is the same state as s' whose cost equals that of s' in CLOSED, remove the same state and put s' into OPEN;

(d) if there is the same state as s' whose cost equals that of s' in OPEN, add s'.paths to the paths of the same state.

6. return to (2).

4.2.3 Optimization

Word graphs tend to have the a larger number of edges originating from the start node than edges originating from another node. Therefore, when D-operator is applied to a state whose node attribute is the start node, many successor states are generated consuming processing time, which is the case when the head of a matching-sequence is a D record. We prepare a series of pseudo edges and nodes originating from the start node to avoid the generation of a large number of successor states. This time, when D-operator is applied to a state whose node is the start node, the state is expanded to a successor state whose node is the first pseudo node. The first pseudo node is the source of edges whose labels are the second words of candidate sentences and the edges flow into the ordinary network. A state of the first pseudo node is expanded into a state of an ordinary node by E-operator or S-operator and into a state of the second pseudo node by D-operator. It can be deduced from the possible maximum distance threshold how many steps of pseudo nodes we should prepare. On the condition that the length of a candidate sentence is L and the length of the series of D records on the head of a matching-sequence is d, the input sentence with the least possible distance is the sentence made from the candidate by deleting d words in the head. Then the distance is $d/((L-d)+L)$. If this distance is greater than the maximum distance threshold Θ, we can give up the search. Therefore, $d/((L-d)+L) \leq \Theta$ is the constraint on d and it deduces $d \leq 2\Theta L/(1+\Theta)$. The maximum integer of d on this condition is the number of steps of pseudo nodes we should prepare.

5 Evaluation

We evaluated the performance of D^3 implemented with the proposed retrieval method through experiments on Japanese-to-English translation in a travel conversation domain using a large-scale corpus.

5.1 Experimental Conditions

We employed a Japanese-and-English parallel corpus, the Basic Travel Expression Corpus (BTEC) (Takezawa and Kikui, 2003). BTEC is a collection of Japanese sentences and their English translations usually found in phrase-books for foreign tourists. The statistics of the corpus are shown in Table 1. For the experiments, a training set of 304,340 sentence pairs and a test set of 510 Japanese sentences were extracted from the corpus. We also used training sets of a half, a quarter, an eighth or a sixteenth the size of the original training set, where a larger set included a smaller set. We call the size of the original set 300K, and others 150K, 75K, 38K and 19K.

Table 1: Statistics of the Corpus

	Japanese	English
# of sentences	404,022	
# of words	2,870,280	2,473,711
avg. sentence length	7.10	6.12
vocabulary size	33,288	22,378

To evaluate translation quality, we employed objective measures and a subjective measure. The objective measures used were the BLEU score (Papineni et al., 2002) and Multi-reference Word Error Rate (mWER) (Ueffing et al., 2002). Sixteen references were used for these measures. Achieving a higher score by BLEU and a lower score by mWER means that the translation results can be regarded as more adequate translations.

For the subjective measure (SM), each translation result was graded into one of four ranks by a bilingual human translator who is a native speaker of the target language, American English. The four ranks were (A) Perfect: no problem in either information or grammar; (B) Fair: easy-to-understand with some unimportant information missing or flawed grammar; (C) Acceptable: broken but understandable with effort; and (D) Nonsense: important information has been translated incorrectly (Sumita et al., 1999). In the experimental results, we present the SM as the cumulative relative frequencies of the evaluation ranks: A, AB and ABC. ABCD is shown as an output rate.

We used thesauri whose hierarchies are based on the Kadokawa Ruigo-shin-jiten (Ohno and Hamanishi, 1984), to calculate the semantic distances. We used a personal computer with Pentium4/2GHz and Allegro Common Lisp 6.2.

Table 2: Training Corpus Size and Performance (θ is the distance threshold)

θ	size	mWER	BLEU	SM (%)			output	time (msec)	
				A	AB	ABC	rate (%)	avg.	max.
1/3	19K	0.4596	0.5448	53.7	65.9	71.0	82.5	62	550
	38K	0.4108	0.5967	58.2	70.6	75.1	86.3	85	930
	75K	0.3833	0.6295	64.7	74.1	79.0	88.8	121	1,690
	150K	0.3401	0.6554	71.2	80.4	83.3	91.8	218	3,310
	300K	0.3198	0.6508	71.2	81.8	84.5	93.3	320	6,650
1/4	19K	0.5060	0.4852	51.0	61.6	64.1	72.2	31	390
	38K	0.4583	0.5579	55.1	65.9	68.8	77.6	41	680
	75K	0.4150	0.6413	63.1	71.8	75.1	82.2	53	530
	150K	0.3602	0.6775	70.2	78.4	80.2	86.5	96	1,600
	300K	0.3349	0.6678	70.4	80.2	82.2	89.4	133	2,040

5.2 Performance

Table 2 shows the evaluation of translation results using the distance thresholds (θ) of 1/3 and 1/4. This table displays the relationship between training corpus size and performance, i.e., translation quality, output rate and processing time. Translation quality increases as the corpus size increases, where the subjective measure and objective measures roughly correspond to each other. When using the lower distance threshold, the processing time is clearly shorter although the output rate naturally decreases. Under three conditions, i.e., the conditions of using the distance threshold of 1/3 and the 150K or 300K corpus, or the distance threshold of 1/4 and the 300K corpus, the rate of rank A exceeds 70% and that of AB reaches above 80%. Under all the conditions, the average processing time is less than 0.4 second. Under the condition of using the threshold of 1/4 and the 300K corpus, where the translation quality is high, the average processing time is about 0.1 second and the maximum time is 2 seconds, indicating that the proposed retrieval method achieves efficient processing.

Figure 2 illustrates the relationship between corpus size and average processing time with axes of logarithmic scale. Although the processing time increases as the corpus size increases, the increasing scale is not linear but about a half power of the corpus size.

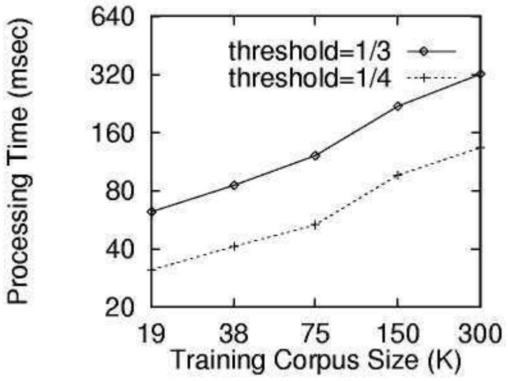

Figure 2: Training Corpus Size and Processing Time

5.3 Comparison with DP-matching

We compared the proposed graph-based retrieval method with naive methods repeating DP-matching. We prepared three methods called Simple-DP, Class-DP and Pruning-DP. Simple-DP uses a hash table to retrieve exactly matched sentences, and if there are no such sentences, repeats DP-matching for an input sentence against all different source sentences in the corpus to find the sentences with the minimum distance. Class-DP improves upon Simple-DP by using the candidate set classification described in Sect. 3.1. Pruning-DP improves upon Class-DP by aborting a DP-matching procedure as soon as the distance between the two concerned sentences is proved to be greater than the minimum distance so far or the threshold. These three methods and the proposed method retrieve the same set of similar sentences for a given input sentence. In the experiment, we executed translations using the retrieval methods and compared their processing time, where the distance threshold used was 1/3.

Table 3 shows the average processing time for each method. Pruning-DP actually improves upon Simple-DP and Class-DP. However, the proposed method far exceeds Pruning-DP. The proposed method is 8.7 times as efficient as Pruning-DP on the 19K corpus and 12.4 times

Table 3: Comparison with DP-matching-based Methods on Average Processing Time (processing time for each method is represented with the unit of milli-second)

corpus size	19K	38K	75K	150K	300K
different sentence#	15,923	29,785	54,657	97,116	199,664
Simple-DP	2,752	4,815	8,101	12,731	26,189
Class-DP	1,286	2,233	3,813	6,045	10,925
Pruning-DP	**539**	**880**	**1,449**	**2,310**	**3,961**
Proposed method	**62**	**85**	**121**	**218**	**320**

on the 300K corpus.

6 Conclusion

We reported on a retrieval method for a sentence-wise EBMT system using edit-distance, and the evaluation of its performance using a large-scale corpus. In experiments for performance evaluation, we used a bilingual corpus comprising hundreds of thousands of sentences from a travel conversation domain. Experimental results show that the EBMT system achieved a high-quality translation ability by using a large corpus, and also achieved efficient processing by using the proposed retrieval method.

Acknowledgements

The authors' heartfelt thanks go to Kadokawa-Shoten for providing the Ruigo-Shin-Jiten. This research was supported in part by the National Institute of Information and Communications Technology.

References

T. Baldwin and H. Tanaka. 2001. Balancing up efficiency and accuracy in translation retrieval. *Journal of Natural Language Processing*, 8(2):19–37.

J. A. Brzozowski. 1962. Canonical regular expressions and minimal state graphs for definite events. *Proc. of Symposium of Mathematical Theory of Automata, MRI Symposia Series*, 12:529–561.

H. T. Cormen, C. E. Leiserson, and L. R. Rivest. 1989. *Introduction to Algorithms.* The MIT Press, London.

L. Cranias, H. Papageorgiou, and S. Piperidis. 1997. Example retrieval from a translation memory. *Natural Language Engineering*, 3(4):255–277.

N. Nilsson. 1971. *Problem-Solving Methods in Artificial Intelligence.* McGraw-Hill, New York.

S. Ohno and M. Hamanishi. 1984. *Ruigo-Shin-Jiten (in Japanese).* Kadokawa, Tokyo, Japan.

K. Papineni, S. Roukos, T. Ward, and W. Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. *Proc. of 40th Annual Meeting of ACL*, pages 311–318.

E. Planas and O. Furuse. 1999. Formalizing translation memories. *Proc. of 7th MT Summit*, pages 331–339.

S. Sato. 1992. CTM: An example-based translation aid system. *Proc. of COLING '92*, pages 1259–1263.

H. Somers. 2003. An overview of ebmt. In M. Carl and A. Way, editors, *Recent Advances in Example-Based Machine Translation*, pages 3–57. Kluwer Academic Publishers, Boston/Dordrecht/London.

E. Sumita and H. Iida. 1991. Experiments and prospects of example-based machine translation. *Proc. of 29th Annual Meeting of ACL*, pages 185–192.

E. Sumita, S. Yamada, K. Yamamoto, M. Paul, H. Kashioka, K. Ishikawa, and S. Shirai. 1999. Solutions to problems inherent in spoken-language translation: The ATR-MATRIX approach. *Proc. of 7th MT Summit*, pages 229–235.

E. Sumita. 2003. An example-based machine translation system using DP-matching between word sequences. In M. Carl and A. Way, editors, *Recent Advances in Example-Based Machine Translation*, pages 189–209. Kluwer Academic Publishers, Boston/Dordrecht/London.

T. Takezawa and G. Kikui. 2003. Collecting machine-translation-aided bilingual dialogues for corpus-based speech translation. *Proc. of EUROSPEECH*, pages 2757–2760.

N. Ueffing, F.J. Och, and H. Ney. 2002. Generation of word graphs in statistical machine translation. *Proc. of Conf. on Empirical Methods for Natural Language Processing*, pages 156–163.