The Relationship Between Architectures and Example-Retrieval Times

Eiichiro SUMITA, Naoya NISIYAMA and Hitoshi IIDA
ATR Interpreting Telecommunications Research Laboratories
2-2 Hikaridai, Seika, Souraku, Kyoto 619-02, JAPAN
sumita@itl.atr.co.jp

Abstract

This paper proposes a method to find the most suitable architecture for a given response time requirement for Example-Retrieval (ER), which searches for the best match from a bulk collection of lingusitic examples. In the Example-Based Approach(EBA), which attains substantially higher accuracy than traditional approaches, ER is extensively used to carry out natural language processing tasks, e.g., parsing and translation. ER, however, is so computationally demanding that it often takes up most of the total sentence processing time. This paper compares several accelerations of ER on different architectures, i.e., serial, MIMD and SIMD. Experimental results reveal the relationship between architectures and response times, which will allows us to find the most suitable architecture for a given response time requirement.

Introduction

Novel models for natural language processing (NLP) that make use of the increasing availability of large-scale corpora and bulk processing power have been studied in recent years. They are called Example-Based Approaches (EBAs) because they rely upon linguistic examples, such as translation pairs, derived from corpora.

Example-Retrieval (ER), the central mechanism of the EBA, provides the best match, which improves the coverage and accuracy of NLP systems over those of traditional methods. ER, however, is computationally demanding. Success in speeding up ER using a massively parallel associative memory processor has already been reported. This paper does not focus on such a single implementation but aims to clarify the relationship between architectures and response times, and answers the question, **which architecture is most suitable for ER?**

First, EBA and ER are briefly introduced, then, the computational cost of ER is described, next, various implementations and experimental results are explained, followed by discussion.

Example-Based Approach

In the early 1980s, Nagao proposed a novel model for machine translation, one that translates by mimicking best-match translation examples, based on the fact that a human translates according to past translation experience.(Nagao 1984) Since the end of the 1980s, large corpora and powerful computational devices have allowed us to realize Nagao's model and expand the model to deal with not only translation but also other tasks such as parsing. First, the word selection problem in machine translation was attacked (Sato 1991; Sumita & Iida 1991; Nomiyama 1992), then, experimental full translation systems [1] were realized or proposed (Sato 1991; Furuse & Iida 1992; Kitano 1991; Watanabe 1992; Maruyama & Watanabe 1992), next, case frame selection (Nagao 1992) and pp-attachment(Sumita, Furuse & Iida 1993) were investigated. These papers have demonstrated that EBAs surpass conventional approaches in several aspects, particularly coverage and accuracy.

So far, many different procedures for ER have been proposed, however, they share a framework that calculates the semantic distance between the input expression and the example expression and returns the examples whose semantic distance is minimum. Here, a typical definition of semantic distance measure is explained(Sumita & Iida 1991). Suppose Input **I** and Example **E** are composed of n words. The semantic distance measure is the summation of the distance, $d(I_k, E_k)$ at the k-th word, multiplied by the weight of the k-th word, w_k. The distance, $d(I_k, E_k)$ is determined based on a thesaurus. The weight, w_k, is the degree to which the word influences the task.

As explained in detail in the next section, ER is computationally demanding. This problem was attacked through the Massively Parallel Artificial Intelligence (MPAI) paradigm(Kitano et al. 1991; Stanfill & Waltz 1986), producing a successful result(Sumita

[1] Translation examples are varied from phrases to sentences. Some input sentences match a whole example sentence. Other input sentences match a combination of fragmental examples (noun phrase, verb phrase, adverbial phrase and so on).

et al. 1993) using a massivley parallel associative processor, IXM2(Higuchi et al. 1991). Unlike this, the authors aim to obtain the relationship between various architectures and response times.

The Computational Cost of Example-Retrieval

ER is the full retrieval of similar examples from a large-scale example database. The total ER time is predominant in processing a sentence and depends on two parameters, i.e., the example database size and the input sentence length. From an estimation of the two parameters, we can derive the time requirement.

Two Parameters

The ER time, T, rises according to the number of examples, N. N is so large, as explained in the next subsection that T is considerable. ER is called many times while processing a sentence. The number of ER calls, C, rises according to the sentence length, L. Consequently, the total ER time is the predominant factor in the time required for processing a sentence by an EBA system. For example, in Furuse et al.'s prototype system for spoken language translation, ER takes up about 50-80% of the time required for translating a sentence and other processes such as pattern-matching, morphological analysis and generation consuming the rest of the time.(Oi et al. 1993) If N increases, the ER/other ratio will increase.

Response Time Requirement

Here, we estimate N and L by extrapolating those of the machine translation system mentioned at the end of the previous subsection.

N depends on the vocabulary size. The vocabulary size of the prototype system is about 1,500. The vocabulary size of the average commercially available machine translation system is about 100,000. In the prototype, N of the most common example is about 1,000. N, in direct proportion to the vocabulary size, 100,000 is about 70,000. For the sake of convenience, we assume $N = 100,000$.

The investigation of the ATR dialogue database (Ehara, Ogura & Morimoto 1990) reveals that the lengths of most sentences are under 20 (words), i.e., $L = 20$. In our experiments, the following approximate equation holds between the number of ER calls, C and the sentence length, L.

$$C = 10^{\frac{L}{10}} \quad (1)$$

In sum, $N = 100,000$ and $L = 20$, i.e., $C = 100$. Then the total ER time with a serial machine is so large that it would be not acceptable in a real-time application, such as interpreting telephony, which we are striving to realize.

Goal: an ER time, T, with 100,000 examples under 1 millisecond.

Achieving this goal means that we will have succeeded in accelerating ER to a sufficient speed because the maximum of the total ER time, 100 (= T*C = 1*100) milliseconds is much less than an utterence time.

Implementations of ER

This section explains items related to implementations: architectures, the task, example and thesaurus-based calculation of semantic distance, and the retrieval algorithms.

Architectures

Although we have implemented and are implementing ER on many machines [2], this paper concentrates on the DEC alpha/7000, KSR1(Kendall Square Research Corp.1992) and MP-2(MasPar Computer Corp. 1992) as representative of three architectures - serial, MIMD and SIMD - respectively.

Task, Example and Thesaurus-Based Calculation of Semantic Distance

For the experiments, we chose the task of translating Japanese noun phrases of the form"A の B" into English (A and B are Japanese nouns. "の" is an adnominal particle such as "の," "で の," "から の," and so on.) They are translated into various English noun phrases of the form "$\tilde{B}$ of $\tilde{A}$," "$\tilde{B}$ in $\tilde{A}$," "$\tilde{B}$ for $\tilde{A}$," "$\tilde{B}$ at $\tilde{A}$," and so on ($\tilde{A}$ and $\tilde{B}$ being English translations of A and B.) We used 100,000 examples, which were collected from the ASAHI newspaper(Tanaka 1991).

"A の B" is represented in the strucure that consists of the fields: AW for the WORD of A; BW for the WORD of B; AC for the the CODE (thesaurus code explained below) of A; BC for the the CODE of B; and NO for the WORD of "の". The example database is stored in an array of the structure with another field, D, for the semantic distance.

Each word corresponds to its concept in the thesaurus. The semantic distance between words is reduced to the semantic distance between concepts. The semantic distance between concepts is determined according to their positions in the the thesaurus hierarchy [3], which is a tree. The semantic distance varies from 0 to 1. When the thesaurus is $(n+1)$-layered, the semantic distance, (k/n) is given to the concepts in the k-th layer from the bottom ($0 \leq k \leq n$).

The semantic distance is calculated based on the CODE (thesaurus code), which clearly represents the thesaurus hierarchy, as in Table 1, instead of traversing

[2] They include the CM-2(Thinking Machines Corp. 1990), IXM2(Higuchi et al. 1991), iPSC/2(Intel 1989), CM-5(Thinking Machines Corp. 1991), and a workstation cluster.

[3] The hierarchy is in accordance with a general thesaurus (Ohno & Hamanishi 1984),a brief explanation of which is found in the literature(Sumita & Iida 1992).

the hierarchy. Our n is 3 and the width of each layer is 10. Thus, each word is assigned a three-digit decimal code of the concept to which the word corresponds.

Table 1: **Thesaurus-Based Calculation of Semantic Distance** - The input CODE and example CODE are $CI = CI_1CI_2CI_3$, $CE = CE_1CE_2CE_3$ -

Condition	Example	Dist.
$CI_1CI_2CI_3 = CE_1CE_2CE_3$	347 , 347	0
$CI_1CI_2 = CE_1CE_2, CI_3 \neq CE_3$	347 , 346	1/3
$CI_1 = CE_1, CI_2 \neq CE_2$	347 , 337	2/3
$CI_1 \neq CE_1$	347 , 247	1

Retrieval Algorithms

Here, we explain two alogorithms.

The Basic Retrieval Algorithm This is used to find the minimum-distance examples by calculating the semantic distance between an input and every example, i.e., an exhaustive search. The algorithm consists of three steps: WORD Exact Match; CODE Exact Match and CODE Partial Match. These include two parts, i.e., a mark part and a collection part, as follows:

1. **[WORD Exact Match]**
   (a) [**Mark**]Examples whose AW, NO, BW match the input are marked.
   (b) [**Collection**]If there is a match, all marked examples and the distance 0 are returned.
2. **[CODE Exact Match]**
   (a) [**Mark**]Examples whose AC, NO, BC match the input are marked.
   (b) [**Collection**]If there is a match, all marked examples and the distance 0 are returned.
3. **[CODE Partial Match]**
   (a) [**Mark**] First, the distance between NOs [4] multiplied by the weight of NO is added to the field, D of the examples.
   Second, the distance between ACs is calculated according to Table 1, and the distance multiplied by the weight is added to the field, D of examples.
   Third, the distance between BCs is calculated according to Table 1, and the distance multiplied by the weight is added to the field, D of examples.
   (b) [**Collection**]All the minimum-distance examples and the minimum distance are returned.

[4]The distance between NOs is 0 or 1 depending on whether or not they match. An example that does NOT match the NO is helpful for translation when the NO of the input has a restricted meaning and the NO of the example has a general meaning.

The Index-Based Retrieval Algorithm The basic algorithm is speeded up by an indexing technique for suppressing unnecessary computation. Step 1 is easily speeded up by hashing of AW, NO, BW. Step 2 is easily speeded up by hashing of AC, NO, BC as well. Calculation between thesaurus codes, AC or BC, is the predominant part of Step 3. We use thesaurus codes as an index of the example array. According to Table 1, if $CI_1 \neq CE_1$ (most of the examples), we need not compute the distance between the input and the example because it is always 1, otherwise, we need to check the example in more detail to compute the distancee between the input and the example. Thus, indexing is useful for accelerating the retrieval by suppressing unnecessary computation.

Experimental Results

The following subsections describe implementations on three different architecutures - serial, MIMD and SIMD - one by one.

Serial Machine

Figure 1 shows serial implementations: (1) the basic retrieval algorithm and (2) the index-based retrieval algorithm, which were explained in the previous section.

Because the two algorithms are realized by repetition, the processing time rises directly according to the number of examples. But (2) drastically improves the response time of (1).

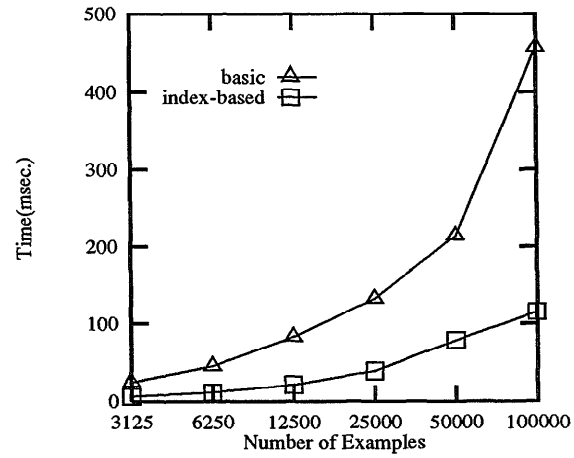

Figure 1: **Time vs. Number of Examples on a Serial Machine, DEC alpha/7000** - *Basic vs. Index-based algorithms* -

Serial machines, unlike parallel machines, have no communication overhead. The ER time, T (msec.) is inversely proportional to the performance of the pro-

cessor, M (MIPS):

$$T = \frac{m}{M} \quad (2)$$

Implementations on seven different machines (Figure 2) revealed that the constant m is about 20,000.

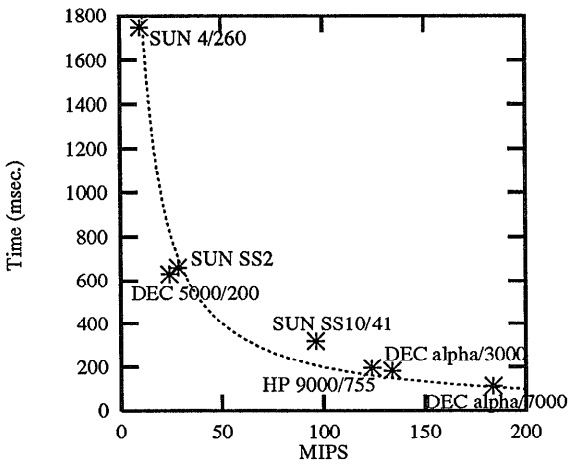

Figure 2: **Time vs. MIPS on Seven Serial machines** - *Index-based algorithm with 100,000 examples* -

MIMD Machine

The flow of MIMD implementation is simple:

1. Divide the example database into equal subdatabases in advance.
2. Execute a serial retrieval algorithm for each processor's subdatabase in parallel.
3. Merge the results of all processors.

In step 2, we can achieve both (1) serial speedup by algorithmic inventions such as indexing and increase of machine performance (MIPS) and (2) parallel speedup by decreasing the example database size to 1/p (p is the number of processors).

Step 3 is the major obstacle to parallel acceleration. Suppose Tp is the response time using p ($p \geq 1$) processors. Tp is the sum of the retrieval time (Step 2), $T1/p$ and the communication time (Step 3), Cp.

$$Tp = \frac{T1}{p} + Cp \quad (3)$$

Cp depends on communication hardware and software models. For example, in a multicomputer, the iPSC/2, the hardware is a hypercube and the software model is a message passing. In such a message passing system, centralizing the message in a single processor is expensive and does not permit scalability. Parallel merging is a common countermeasure(Ranka & Sahni 1990; Sato 1993). However, the iPSC/2 showed a poor response time, because although the Cp of a parallel merging algorithm is $c * log_2 p$, unfortunately the constant c is considerable, about 27.5 milliseconds. In a multiprocessor, the KSR1, the hardware is a directory and the programming model is a shared memory model using pthread, mutex(lock/unlock) and barrier. Figure 3 demonstrates that the speedup of ER on the KSR1 is good because Cp is small.

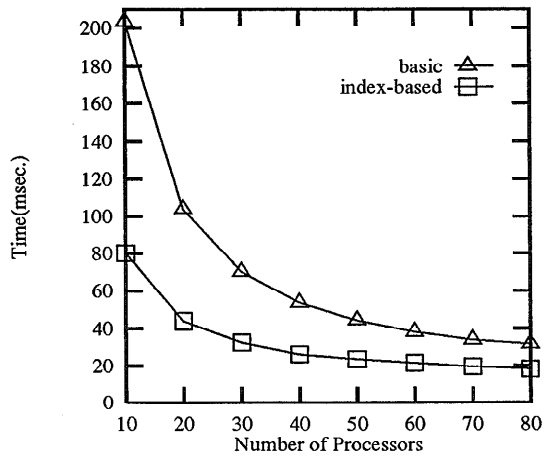

Figure 3: **Time vs. Number of Processors on a MIMD machine, KSR1** - *Basic vs. Index-based algorithms with 100,000 examples* -

SIMD Machine

A SIMD machine can efficiently execute the basic algorithm because (1) a minimal time is required to compute the semantic distance because computation is completely paralel within $N \leq p$, and p is larger than that of MIMD machines at an order of magnitude. If $N > p$, then a serial repetition must emerge; however, the degree is drastically smaller than with MIMD machines. (2) low overhead required to merge the results. (3) it can benefit from indexing technique(Figure 4).

If $N \leq p$, i.e., each processor has, at most, one example, the computational cost is not o(N) but o(1). because there is no speedup or slowdown due to a processor that works in vain. If $N > p$, then the example arrays are folded to fit p processors and the same operation is repeated $(N-1)/p + 1$ times. Thus, the response time of SIMD implementation rises stepwise according to the number of examples.

The Best Speedup Method

This section discusses accelerating strategies from a baseline machine, a serial-virtual machine (40 MIPS), to meet the goal, **100,000 examples per millisecond** based on the results for three state-of-the-art machines, DEC alpha/7000, KSR1 and MP-2.

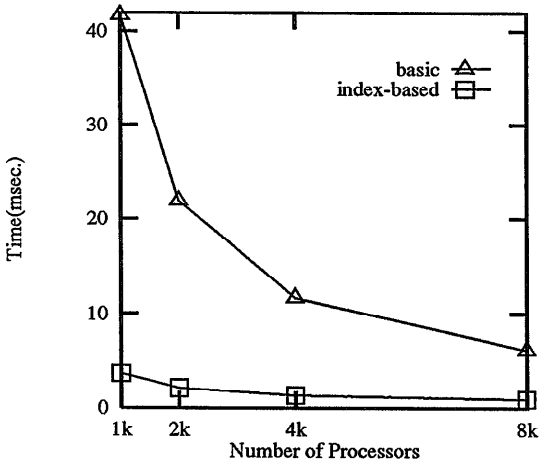

Figure 4: **Time vs. Number of Processors on a SIMD machine, MP-2** - *Basic and index-based algorithms with 100,000 examples* -

There are four strategies for improving the response time to achieve our goal:

1. (Serial Machine) Increase the performance of processor, M (MIPS).
2. (MIMD Machine) Increase the number of processors, p of the same performance, 40 MIPS.
3. (MIMD Machine) Increase both the number of processors, p and the performance of each processor, M (MIPS).
4. (SIMD Machine) Increase the number of processors, p drastically at the expense of the performance of each processor, M (MIPS).

Table 2 summerizes effects of four strategies. Strategy 1 seems to hit a wall [5] because the slope is already very small at 200 MIPS (Figure 2). Strategy 2, i.e., only the increase in p cannot go beyond the overhead, because the slope is already very small at 80 processors (Figure 3). Strategy 3 is feasible if the overhead can be decreased approximately in inverse proportion to the increase in M. [6] Strategy 4, i.e., **SIMD has achieved our goal.** [7]

[5]To attain the goal, we should have 20,000 MIPS according to equation (2). Increasing M to 20,000 MIPS is hopeless because it will take considerable time as explained below. MIPS is increasing at the rate of about 35% per year(Hennessy & Patterson 1990) and we already have a 200-MIPS microprocessor, thus we can get an over 20,000-MIPS processor by 2010, 16 years from now.

[6]The next generation of MIMD machines will meet the requirement.

[7]There has been a question, which came from an implementation using C* on CM-5, about the effectiveness of SIMD for ER.(Sato 1993) We also have experimental evidence that C* on CM-5 is not particularly efficient. Because, in principle, the communication cost of MIMD machines is much higher than that of SIMD machines, SIMD simulation by MIMD is expensive. Our results on the MP-2, however, are counterevidence against the negative question.

EBA systems do not consist of ER only. There are other processes that are suitable for MIMD, while ER is the best fit for SIMD. In such a **heterogeneous** system, if communication between a SIMD machine and a MIMD machine cannot attain sufficient speed, we should adopt strategy 3 rather than strategy 4.

Table 2: **Summary of Four Strategies for Acceleration of ER.** *The serial-virtual machine (40MIPS) and MIMD-virtual machine (200MIPS), which are marked by (*), are virtual machines for discussion and the figures shown are estimated. All figures here are rounded to simplify the comparison.* -

Machine	M	p	$M*p$	T
serial-virtual*	40	1	40	500
1.DEC alpha/7000	200	1	200	100
2.KSR1	40	80	3200	20
3.MIMD-virtual*	200	80	16000	4
4.MP-2	**4**	**8000**	**32000**	**1**

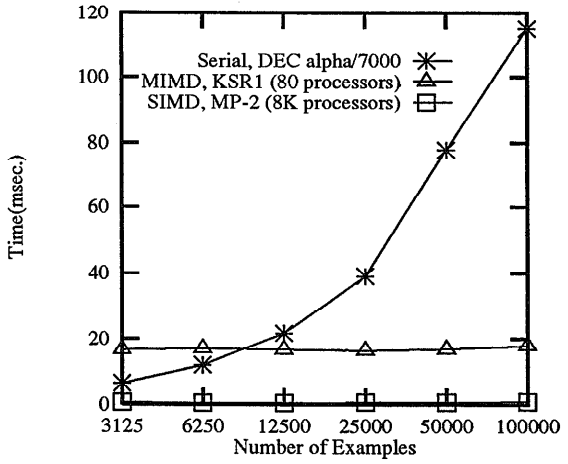

Figure 5: **Time vs. Number of Examples on Serial, MIMD and SIMD Machines** - *Index-based algorithm* -

As shown in Figure 5, SIMD is always the winner of the competition. Note the intersection of the DEC alpha/7000 and KSR1 lines in Figure 5. To the left of the intersection, the serial machine outperforms the MIMD machine due to its zero overhead for parallelization. To the right of the intersection, the MIMD machine out-

performs the serial machine due to the effect of dividing the example database into 1/p subdatabases.

Conclusion

We have implemented Example-Retrieval (ER), which is the key technique of Example-Based Approaches (EBAs), on machines of different architectures such as DEC alpha/7000, KSR1 and MP-2. EBA is a novel approach, which is now being developed energetically because it is expected to overcome the problems of traditional approaches for Natural Language Processing (NLP). The viability of EBA depends on whether we can speed up ER sufficiently. Using the relationship between the architectures and response times derived from experimental results, we have reached the conclusion that SIMD machines outperform machines of other types.

EBA systems, in general, consist of several different processes, which should run best on different architectures. In other words, EBA systems are heterogeneous, thus, we will further pursue the best combination of architectures, interconnects and communication models.

References

Ehara, T., Ogura, K. and Morimoto, T. 1990. ATR Dialogue Database. In Proceedings of ICSLP-90, vol. 2, 1093-1096.

Furuse, O. and Iida, H. 1992. Cooperation between Transfer and Analysis in Example-Based Framework. In Proceedings of Coling-92, 645-651. Nantes.

Hennessy, J. and Patterson, D. 1990. *Computer Architecture A Quantitative Approach.* Morgan Kaufmann Publishers.

Higuchi, T., Kitano, H., Handa, K., Furuya, T., Takahashi, N. and Kokubu, A. 1991. IXM2 A Parallel Associative Processor for Knowledge Processing, In Proceedings of AAAI-91. Anaheim.

Intel Scientific Computers 1989. *iPSC/2 User's Guide.*

Kendall Square Research Corp. 1992. *KSR1 Technical Summary.*

Kitano, H. 1991. ΦDM-Dialog An Experimental Speech-to-Speech Dialog Translation System. *IEEE Computer, June: 36-50.*

Kitano, H., Hendler, J., Moldovan, D., Higuchi, T. and Waltz, D. 1991. Massively Parallel Artificial Intelligence. In Proceedings of IJCAI-91, 557-562.

Maruyama, H. and Watanabe, H. 1992. Tree Cover Search Algorithm for Example-Based Translation. In Proceedings of TMI-92, 173-184.

MasPar Computer Corp. 1992. *The Design of the MasPar MP-2 A Cost Effective Massively Parallel Computer.*

Nagao, M. 1984. A Framework of a Mechanical Translation between Japanese and English by Analogy Principle. In *Artificial and Human Intelligence* eds. A. Elithorn and R. Banerji, North-Holland, 173-180.

Nagao, M. 1992. Some Rationales and Methodologies for Example-Based Approach. In Proceedings of FGNLP-92, 82-94.

Nomiyama, H. 1992. Machine Translation by Case Generalization. In Proceedings of Coling-92, 714-720. Nantes.

Ohno, S. and Hamanishi, M. 1984. *Ruigo-Shin-Jiten.* Kadokawa.

Oi, K., Sumita, E. Furuse, O., Iida, H. and Kitano, H. 1993. Toward Massively Parallel Spoken Language Translation. In Proceedings of PPAI-93 Workshop (IJCAI-93), 36-39.

Ranka, S. and Sahni, S. 1990. *Hypercube Algorithms.* Springer-Verlag.

Sato, S. 1991. Example-Based Machine Translation. Ph.D. Diss., Dept. of Electrical Engineering, Kyoto University.

Sato, S. 1993. MIMD Implementation of MBT3. In Proceedings of PPAI-93 Workshop (IJCAI-93), 28-35.

Stanfill, C. and Waltz, D. 1986. Toward Memory-Based Reasoning, CACM Vol. 29, No. 12, 1213-1228.

Sumita, E. and Iida, H. 1991. EXPERIMENTS AND PROSPECTS OF EXAMPLE-BASED MACHINE TRANSLATION. In Proceedings of 29th ACL, 185-192.

Sumita, E. and Iida, H. 1992. Example-Based Transfer of Japanese Adnominal Particles into English. *IEICE TRANS. INF. & SYST. Vol.E75-D, No.4, 585-594.*

Sumita, E., Furuse, O. and Iida, H. 1993. An Example-Based Disambiguation of Preposiotinal Phrase Attachment. In Proceedings of TMI-93, 80-91.

Sumita, E., Oi, K., Furuse, O., Iida, H., Higuchi, T., Takahashi, N. and Kitano, H. 1993. Example-Based Machine Translation on Massively Parallel Processors. In Proceedings of IJCAI-93, 1283-1288.

Tanaka, Y. 1991. *GO-TO-GO-NO-KANKEI-KAISEKIYOU-SHIRYOU, NO-WO-CYUSHIN-TO-SHITA.* AICHI SYUKUTOKU University.

Thinking Machines Corp. 1990. *Model CM-2 Technical Summary.*

Thinking Machines Corp. 1991. *Model CM-5 Technical Summary.*

Watanabe, H. 1992. A Similarity-Driven Transfer System. In Proceedings of Coling-92, 770-776. Nantes.