Real-Time Spoken Language Translation Using Associative Processors

Kozo Oi, Eiichiro Sumita, Osamu Furuse, Hitoshi Iida and **Tetsuya Higuchi†**

ATR Interpreting Telecommunications Research Laboratories
2-2 Hikaridai, Seika, Souraku, Kyoto 619-02, JAPAN
{oi,sumita,furuse,iida}@itl.atr.co.jp

†Electrotechnical Laboratory
1-1-4 Umezono, Tsukuba, Ibaraki 305, Japan
higuchi@etl.go.jp

Abstract

This paper proposes a model using associative processors (APs) for real-time spoken language translation. Spoken language translation requires (1) an accurate translation and (2) a real-time response. We have already proposed a model, TDMT (Transfer-Driven Machine Translation), that translates a sentence utilizing examples effectively and performs accurate structural disambiguation and target word selection. This paper will concentrate on the second requirement. In TDMT, example-retrieval (ER), i.e., retrieving examples most similar to an input expression, is the most dominant part of the total processing time. Our study has concluded that we only need to implement the ER for expressions including a frequent word on APs. Experimental results show that the ER can be drastically speeded up. Moreover, a study on communications between APs demonstrates the scalability against vocabulary size by extrapolation. Thus, our model, TDMT on APs, meets the vital requirements of spoken language translation.

1 Introduction

Research on speech translation that began in the mid-1980s has been challenging. Such research has resulted in several prototype systems (Morimoto et al., 1993; Kitano, 1991; Waibel et al., 1991). Speech translation consists of a sequence of processes, i.e., speech recognition, spoken language translation and speech synthesis. Each process must be accelerated in order to achieve real-time response. This paper focuses on the second process, spoken language translation, which requires (1) an accurate translation and (2) a real-time response. We have already proposed a model that utilizes examples and translates a sentence by combining pieces of transfer knowledge, i.e., target language expressions that correspond to source language expressions that cover the sentence jointly. The model is called Transfer-Driven Machine Translation (TDMT) (Furuse and Iida, 1992; Furuse et al., 1994) (see subsection 2.1 for details). A prototype system of TDMT which translates a Japanese spoken sentence into English, has performed accurate structural disambiguation and target word selection[1].

This paper will focus on the second requirement. First, we will outline TDMT and analyze its computational cost. Second, we will describe the configuration, experimental results and scalability of TDMT on associative processors (APs). Finally, we will touch on related works and conclude.

2 TDMT and its Cost Analysis

2.1 Outline of TDMT

In TDMT, transfer knowledge is the primary knowledge, which is described by an example-based framework (Nagao, 1984). A piece of transfer knowledge describes the correspondence between source language expressions (SEs) and target language expressions (TEs) as follows, to preserve the translational equivalence:

$$SE \Rightarrow \begin{array}{ll} TE_1 & (E_{11}, E_{12}, \ldots), \\ \vdots & \vdots \\ TE_n & (E_{n1}, E_{n2}, \ldots) \end{array}$$

E_{ij} indicates the j-th example of TE_i. For example, the transfer knowledge for source expression "X no Y" is described as follows[2]:

X no Y =>
Y' of X' ((ronbun[paper],daimoku[title]),$\cdots$),
Y' for X' ((hoteru[hotel],yoyaku[reservation]),$\cdots$),
Y' in X' ((Kyouto[Kyoto],kaigi[conference]),$\cdots$),
$\vdots$ $\vdots$

[1] The translation success rate for 825 sentences used as learning data in a conference registration task, is about 98%. The translation success rate for 1,056 sentences, amassed through arbitary inputs in the same domain, is about 71%. The translation success rate increases as the number of examples increases.

[2] X and Y are variables for Japanese words and X' and Y' are the English translations of X and Y, respectively; "no" is an adnominal particle that corresponds to such English prepositions as "of," "for," "in," and so on.

TDMT utilizes the semantic distance calculation proposed by Sumita and Iida (Sumita and Iida, 1992). Let us suppose that an input, I, and each example, E_{ij}, consist of t words as follows:

$$I = (I_1, \cdots, I_t)$$
$$E_{ij} = (E_{ij1}, \cdots, E_{ijt})$$

Then, the distance between I and E_{ij} is calculated as follows:

$$d(I, E_{ij}) = d((I_1, \cdots, I_t), (E_{ij1}, \cdots, E_{ijt})) = \sum_{k=1}^{t} d(I_k, E_{ijk}) \times W_k$$

The semantic distance $d(I_k, E_{ijk})$ between words is reduced to the distance between concepts in a thesaurus (see subsection 3.2 for details). The weight W_k is the degree to which the word influences the selection of the translation[3].

The flow of selecting the most plausible TE is as follows:

(1) The distance from the input is calculated for all examples.
(2) The example with the minimum distance from the input is chosen.
(3) The corresponding TE of the chosen example is extracted.

Processes (1) and (2) are called ER (Example-Retrieval) hereafter.

Now, we can explain the top-level TDMT algorithm:

(a) Apply the transfer knowledge to an input sentence and produce possible source structures in which SEs of the transfer knowledge are combined.
(b) Transfer all SEs of the source structures to the most appropriate TEs by the processes (1)–(3) above, to produce the target structures.
(c) Select the most appropriate target structure from among all target structures on the basis of the total semantic distance.

For example, the source structure of the following Japanese sentence is represented by a combination of SEs with forms such as (X no Y), (X ni Y), (X de Y), (X ga Y) and so on:

dainihan	**no**	annaisyo	**ni**
{ second version,	particle,	announcement,	particle,
kaigi	**de**	happyou-sareru	ronbun
conference,	particle,	be presented,	paper,
no	daimoku	**ga**	notte-orimasu
particle,	title,	particle,	be written }

2.2 The Analysis of Computational Cost

Here, we briefly investigate the TDMT processing time on sequential machines.

For 746 test sentences (average sentence length: about 10 words) comprising representative Japanese sentences[4] in a conference registration task, the average translation time per sentence is about 3.53 seconds in the TDMT prototype on a sequential machine (SPARCstation2). ER is embedded as a subroutine call and is called many times during the translation of one sentence. The average number of ER calls per sentence is about 9.5. Figure 1 shows rates for the ER time and other processing time. The longer the total processing time, the higher the rate for the ER time; the rate rises from about 43% to about 85%. The average rate is 71%. Thus, ER is the most dominant part of the total processing time.

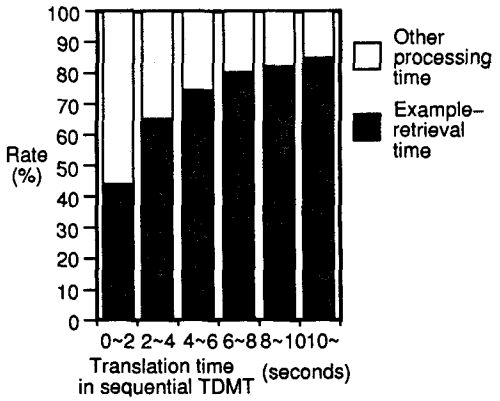

Figure 1: Rates for ER time in sequential TDMT

In the ATR dialogue database (Ehara et al., 1990), which contains about 13,000 sentences for a conference registration task, the average sentence length is about 14 words. We therefore assume in the remainder of this subsection and subsection 3.5 that the average sentence length of a Japanese spoken sentence is 14 words, and use statistics for 14-word sentences when calculating the times of a large-vocabulary TDMT system. The expected translation time of each 14-word sentence is about 5.95 seconds, which is much larger than the utterance time. The expected number of ER calls for each 14-word sentence is about 15. The expected time and rate for ER of the 14-word sentence are about 4.32 seconds and about 73%, respectively.

Here, we will consider whether a large-vocabulary TDMT system can attain a real-time response. In the TDMT prototype, the vocabulary size and the number of examples, N, are about 1,500 and 12,500, respectively. N depends on the vocabulary size. The vocabulary size of the average commercially-available machine translation system is about 100,000. Thus, in the large-vocabulary sys-

[3] In the TDMT prototype, W_k is $1/t$.

[4] We have 825 test sentences as described in footnote 1 in section 1. These sentences cover basic expressions that are used in Japanese ability tests conducted by the government and Japanese education courses used by many schools for foreigners (Uratani et al., 1992). The sentences were reviewed by Japanese linguists. In the experiments in this paper, we used 746 sentences excluding sentences translated by exact-match.

tem, N is about 830,000 ($\simeq 12,500 \times 100,000/1,500$) in direct proportion to the vocabulary size. For the sake of convenience, we assume $N = 1,000,000$. The ER time is nearly proportional to N due to process (1) described in subsection 2.1. Therefore, the expected translation time of a 14-word sentence in the large-vocabulary system using a SPARCstation2 (28.5 MIPS) is about 347.2 (=[ER time]+[other processing time[5]]=[$4.32 \times 1,000,000/12,500$]+[$5.95 - 4.32$]=345.6+1.63) seconds. ER consumes 99.5% of the translation time.

A 4,000 MIPS sequential machine will be available in 10 years, since MIPS is increasing at a rate of about 35 % per year; we already have a 200 MIPS machine (i.e. DEC alpha/7000). The translation time of the large-vocabulary system with the 4,000 MIPS machine is expected to be about 2.474 ($\simeq 347.2 \times 28.5/4,000$) seconds. Of the time, 2.462 ($\simeq 345.6 \times 28.5/4,000$) seconds will be for ER. Therefore, although the 1500-word TDMT prototype will run quickly on the 4,000 MIPS machine, sequential implementation will not be scalable, in other words, the translation time will still be insufficient for real-time application. Therefore, we have decided to utilize the parallelism of associative processors.

Careful analysis of the computational cost in the sequential TDMT prototype has revealed that the ER for the top 10 SEs (source language expressions) accounts for nearly 96% of the entire ER time. The expected number of ER calls for the top 10 SEs of each 14-word sentence is about 6. Table 1 shows rates of the ER time against each SE in the transfer knowledge. Function words, such as "wa", "no", "o", "ni" and "ga", in the SEs are often used in Japanese sentences. They are polysemous, thus, their translations are complicated. For that reason, the number of examples associated with these SEs is very large. In sum, the computational cost of retrieving examples including function words is proportional to the square of the frequency of the function words. In an English-to-Japanese version of TDMT, the number of examples associated with the SEs, which include function words such as "by", "to" and "of", is very large as well.

With this rationale, we decided to parallelize ER for the top 10 SEs of the Japanese-to-English transfer knowledge.

Table 1: Rates of ER time against each SE

SE			Rate(%)	Accumulative(%)
X	wa	Y	25.20	25.20
X	no	Y	20.60	45.80
X	o	Y	19.61	65.41
X	ni	Y	11.13	76.54
X	ga	Y	8.90	85.44
	⋮		⋮	⋮

[5]This time does not depend on N.

3 TDMT Using Associative Processors

3.1 ER on Associative Processors (APs)

As described in the previous subsection, parallelizing ER is inevitable but promising. Preliminary experiments of ER on a massively parallel associative processor IXM2 (Higuchi et al., 1991a; Higuchi et al., 1991b) have been successful (Sumita et al., 1993). The IXM2 is the first massively parallel associative processor that clearly demonstrates the computing power of a large Associative Memory (AM). The AM not only features storage operations but also logical operations such as retrieving by content. Parallel search and parallel write are particularly important operations. The IXM2 consists of associative processors (APs) and communication processors. Each AP has an AM of 4K words of 40 bits, plus an IMS T801 Transputer (25 MHz).

3.2 Semantic Distance Calculation on APs

As described in subsection 2.1, the semantic distance between words is reduced to the distance between concepts in a thesaurus. The distance between concepts is determined according to their positions in the thesaurus hierarchy. The distance varies from 0 to 1. When the thesaurus is $(n+1)$ layered, (k/n) is connected to the classes in the k-th layer from the bottom ($0 \leq k \leq n$). In Figure 2, n is 3, k is from 0 to 3, and the distance d is 0/3 (=0), 1/3, 2/3 and 3/3 (=1) from the bottom.

The semantic distance is calculated based on the thesaurus code, which clearly represents the thesaurus hierarchy, as in Table 2, instead of traversing 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.

Here, we briefly introduce the semantic distance calculation on an AM (Associative Memory) referring to Figure 3. The input data is 344 which is the

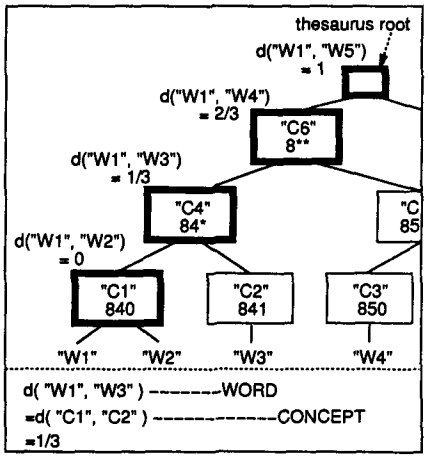

Figure 2: Thesaurus (portion) and distance

Table 2: Semantic distance by thesaurus code. 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

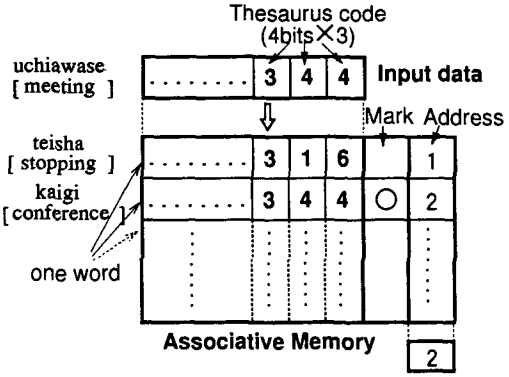

Figure 3: Semantic distance calculation on an Associative Memory

thesaurus code of the word "uchiawase[meeting]". Each code (316, 344) of the examples such as "teisha[stopping]", "kaigi[conference]", and so on is stored in each word of the AM. The algorithm for searching for examples whose distance from the input is 0, is as follows[6]:

(I) Give a command that searches for the words whose three-digit code matches the input. (The search is performed on all words simultaneously and matched words are marked.)

(II) Get the addresses of the matched words one by one and add the distance, 0, to the variable that corresponds to each address.

The search in process (I) is done only by the AM and causes the acceleration of ER. Process (II) is done by a transputer and is a sequential process.

3.3 Configuration of TDMT Using APs

According to the performance analysis in subsection 2.2, we have implemented the ER of the top 10 SEs.

Figure 4 shows a TDMT configuration using APs in which the ER of the top 10 SEs are implemented. The 10 APs ($AP_1, AP_2, \cdots, AP_{10}$) and the transputer (TP) directly connected to the host machine (SPARCstation2) are connected in a tree configuration[7].

[6] An algorithm that searches for examples whose distance from the input is 1/3, 2/3 or 3/3, is similar.

[7] The tree is 3-array because the transputer has four connectors. The TDMT main program is described with Lisp language and is executed on the host machine. The ER routine is programmed with Occam2 language, which is called by the main program and runs on the TP and on transputers in the APs.

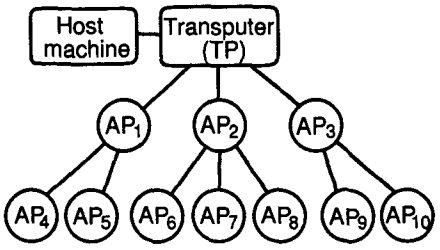

Figure 4: Configuration of TDMT using 10 APs

The algorithm for ER in the TDMT using APs is as follows:

(i) Get input data and send the input data from the host to TP.

(ii) Distribute the input data to all APs.

(iii) Each AP carries out ER, and gets the minimum distance and the example number whose distance is minimum.

(iv) Each AP and the TP receive the data from the lower APs (if they exist), merge them and their own result, and send the merged result upward.

With the configuration shown in Figure 4, we studied two different methods of storing examples. The two methods of storing examples are as follows:

Homo-loading (HM) Examples associated with one SE are stored in one AP. That is, each AP is loaded with examples of the same SE.

Hetero-loading (HT) Examples associated with one SE are divided equally and stored in 10 APs. That is, each AP is loaded with examples of 10 different SEs.

3.4 Experimental Results

Figure 5 plots the speedup of ER for TDMT using APs over sequential TDMT, with the two methods. It can be seen that the speedup for the HT method is greater than that for the HM method, partly because the sequential part of ER is proportional to the example number in question. With the HT method,

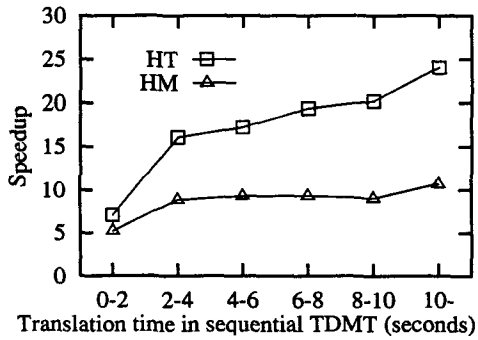

Figure 5: Speedup of ER in TDMT using APs over sequential TDMT

the average speedup is about 16.4 (=[the average time per sentence in the sequential TDMT]/[the average time per sentence in the HT method]≃ 2489.7/152.2(msec.)). For the 14-word sentences, the average speedup is about 20.8 (≃ 4324.7/208.0(msec.)) and the ER time for the top 10 SEs is about 85.4 milliseconds out of the total 208.0 milliseconds.

Figure 6 shows a screen giving a comparison between TDMT using APs and sequential TDMT.

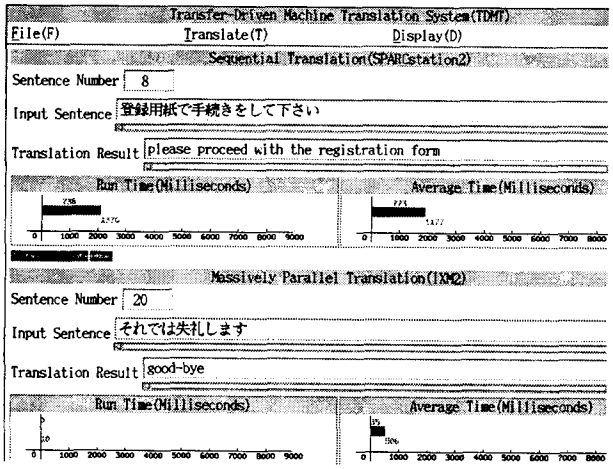

Figure 6: A comparison of TDMT using APs and sequential TDMT — This is a snapshot of a race between two machines. The sentence numbers and run times correspond to sentences that have been translated. The average times cover all sentences that have been translated.

3.5 Scalability

In this subsection, we consider the scalability of TDMT using APs in the HT method. Here, we will estimate the ER time using 1,000,000 examples which are necessary for a large-vocabulary TDMT system (see subsection 2.2).

Assuming that the number of examples in each AP is the same as that in the experiment, 800 (= 1,000,000/12,500) APs are needed to store 1,000,000 examples. Figure 7 shows 800 APs in a tree structure ($\sum_{x=1}^{L} 3^x \geq 800$; L(minimum)=6 layers). In the remainder of this subsection, we will use the statistics (time, etc.) for the 14-word[8] sentences.

The translation time is divided into the ER time on APs and the processing time on the host machine. The former is divided into the computing time on each AP and the communication time between APs.

The ER time on APs in the experiment is about 85.4 milliseconds as described in subsection 3.4. The computing time per sentence on each AP is the same as that in the experiment and is approximately 84.1 milliseconds out of the 85.4 milliseconds. The communication time between APs is vital and increases

[8]This is the average sentence length in the ATR dialogue database. See subsection 2.2.

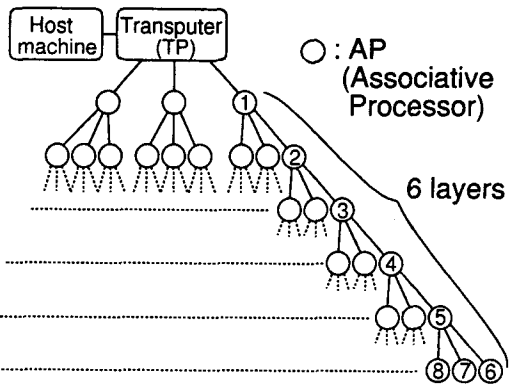

Figure 7: Configuration of large-vocabulary TDMT using 800 APs

as the number of APs increases. There are two kinds of communication processes: distribution of input data[9] and collection of the resulting data of ER[10].

The input data distribution time is the sum of distribution times TP→ AP_1, AP_1 → AP_2, ···, AP_4→AP_5 and AP_5→AP_6, that is, 6 multiplied by the distribution time between two APs that are directly connected (see Figure 7), because a transputer can send the data to the other transputers directly connected in parallel (e.g., AP_5→AP_6, AP_5→AP_7, AP_5→AP_8). The average number of ER calls is about 6 and the average distribution time between directly-connected APs is about 0.05 milliseconds. Therefore, the total input data distribution time per sentence in the configuration of Figure 7 is nearly 1.8 (= 0.05 × 6 × 6) milliseconds.

The time required to collect the resulting data is the sum of the processing times in process (iv), which is explained in subsection 3.3, at the TP, $AP_1, \cdots, AP_4$ and AP_5, illustrated in Figure 7. It takes about 0.04 milliseconds, on average, for each AP to receive the resulting data from the lower APs and it takes about 0.02 milliseconds, on average, for the AP to merge the minimum distance and the example numbers. Therefore, it is expected that the total collection time is about 2.2 (= (0.04 + 0.02) × 6 × 6) milliseconds.

Thus, the total communication time is about 4.0 (= 1.8 + 2.2) milliseconds. Consequently, the processing time on APs is about 88.1 (= 84.1 + 4.0) milliseconds. This is 3,920 (≃ 345.6/0.0881) times faster than the SPARCstation2[11]. It is clear then that the communication has little impact on the scalability because it is controlled by the tree depth and small coefficient.

Therefore, the TDMT using APs becomes more scalable as the number of examples increases and can attain a real-time response.

[9]Process (ii) described in subsection 3.3.

[10]Process (iv) described in subsection 3.3.

[11]See the data described in subsection 2.2.

4 Related works

Up to now, some systems using a massively parallel machine in the field of natural language processing, such as a parsing system (Kitano and Higuchi, 1991b) and translation systems, e.g., Dm-SNAP (Kitano et al., 1991), ASTRAL (Kitano and Higuchi, 1991a), MBT3n (Sato, 1993), have been proposed. They have demonstrated good performance; nonetheless, they differ from our proposal. For the first three systems, their domain is much smaller than our domain and they do not perform structural disambiguation or target word selection based on the semantic distance between an input expression and each example. For the last system, it translates technical terms i.e. noun phrases, but not sentences.

5 Conclusion

This paper has proposed TDMT (Transfer-Driven Machine Translation) on APs (Associative Processors) for real-time spoken language translation. In TDMT, a sentence is translated by combining pieces of transfer knowledge that are associated with examples, i.e., source word sequences. We showed that the ER (example-retrieval) for source expressions including a frequent word, such as a function word, are predominant and are drastically speeded up using APs. That the TDMT using APs is scalable against vocabulary size has also been confirmed by extrapolation, i.e., a 10-AP sustained performance to an 800-AP expected performance, through analysis on communications between APs. Consequently, the TDMT can achieve real-time performance even with a large-vocabulary system. In addition, as our previous papers have shown, the TDMT achieves accurate structural disambiguation and target word selection. Thus, our model, TDMT on APs, meets the vital requirements for real-time spoken language translation.

References

Terumasa Ehara, Kentaro Ogura, and Tsuyoshi Morimoto. 1990. ATR Dialogue Database. In *Proc. of ICSLP'90*, pages 1093–1096, November.

Osamu Furuse and Hitoshi Iida. 1992. Cooperation Between Transfer and Analysis in Example-Based Framework. In *Proc. of COLING'92*, pages 645–651, July.

Osamu Furuse, Eiichiro Sumita, and Hitoshi Iida. 1994. Transfer Driven Machine Translation Utilizing Empirical Knowledge. *Transactions of Information Processing Society of Japan*, 35(3):414–425, March.

Tetsuya Higuchi, Tatsumi Furuya, Kenichi Handa, Naoto Takahashi, Hiroyasu Nishiyama, and Akio Kokubu. 1991a. IXM2 : A Parallel Associative Processor. In *Proc. of the 18th International Symposium on Computer Architecture*, May.

Tetsuya Higuchi, Hiroaki Kitano, Tatsumi Furuya, Ken-ichi Handa, Naoto Takahashi, and Akio Kokubu. 1991b. IXM2 : A Parallel Associative Processor for Knowledge Processing. In *Proc. of AAAI'91*, pages 296–303, July.

Hiroaki Kitano and Tetsuya Higuchi. 1991a. High Performance Memory-Based Translation on IXM2 Massively Parallel Associative Memory Processor. In *Proc. of AAAI'91*, pages 149–154, July.

Hiroaki Kitano and Tetsuya Higuchi. 1991b. Massively Parallel Memory-Based Parsing. In *Proc. of IJCAI'91*, pages 918–924.

Hiroaki Kitano, Dan Moldovan, and Seungho Cha. 1991. High Performance Natural Language Processing on Semantic Network Array Processor. In *Proc. of IJCAI'91*, pages 911–917.

Hiroaki Kitano. 1991. ΦDM-Dialog: An Experimental Speech-to-Speech Dialog Translation System. *IEEE Computer*, 24(6):36–50, June.

Tsuyoshi Morimoto, Toshiyuki Takezawa, Fumihiro Yato, Shigeki Sagayama, Toshihisa Tashiro, Masaaki Nagata, and Akira Kurematsu. 1993. ATR's Speech Translation System: ASURA. In *Proc. of EUROSPEECH'93*, pages 1291–1294, September.

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

Satoshi Sato. 1993. MIMD Implementation of MBT3. In *Proc. of the Second International Workshop on Parallel Processing for Artificial Intelligence*, pages 28–35. IJCAI'93, August.

Eiichiro Sumita and Hitoshi Iida. 1992. Example-Based Transfer of Japanese Adnominal Particles into English. *IEICE TRANS. INF. & SYST.*, E75-D(4):585–594, April.

Eiichiro Sumita, Kozo Oi, Osamu Furuse, Hitoshi Iida, Tetsuya Higuchi, Naoto Takahashi, and Hiroaki Kitano. 1993. Example-Based Machine Translation on Massively Parallel Processors. In *Proc. of IJCAI'93*, pages 1283–1288, August.

Noriyoshi Uratani, Masami Suzuki, Masaaki Nagata, Tsuyoshi Morimoto, Yukinori Takubo, Toshiyuki Sadanobu, and Hajime Narita. 1992. A Function Evaluation Method for Analysis System of Goal-Directed Dialogue. In *IEICE Technical Report*, NLC92-10, July.

Alex Waibel, Ajay N. Jain, Arthur E. McNair, Hiroaki Saito, Alexander G. Hauptmann, and Joe Tebelskis. 1991. JANUS: A Speech-to-speech Translation Using Connectionist and Symbolic Processing Strategies. In *Proc. of ICASSP'91*, pages 793–796, May.