

Example-Based Machine Translation on Massively Parallel Processors

Eiichiro SUMITA, Kozo OI, Osamu FURUSE and Hitoshi IIDA

ATR Interpreting Telephony Research Laboratories

2-2 Hikaridai, Seika, Souraku, Kyoto 619-02, JAPAN

{sumita,oi,furuse,iida}@itl.atr.co.jp

Tetsuya HIGUCHI and Naoto TAKAHASHI

Electrotechnical Laboratory

1-1-4 Umezono, Tsukuba, Ibaraki 305 Japan

{higuchi,ntakahas}@etl.go.jp

Hiroaki KITANO

Carnegie Mellon University

Pittsburgh, PA 15213 U.S.A.

hiroaki@cs.cmu.edu

Abstract

This paper describes an Example-Based Machine Translation (EBMT) system implemented on massively parallel computers. EBMT is a novel approach to machine translation that directly uses translation examples to carry out translation tasks. The authors have shown that EBMT attains substantially higher accuracy than traditional machine translation systems. However, EBMT is so computationally demanding that the system would run very slowly on sequential machines and even on a loosely-coupled multi-processor iPSC/2. We have implemented EBMT on an IXM2 massively parallel associative memory processor and a CM-2 connection machine. Experimental results demonstrate that EBMT on the IXM2 exhibits the best performance and attains a response speed that will suffice for real-time tasks such as interpreting telephony.

1 Introduction

This paper describes an Example-Based Machine Translation (EBMT) system implemented on massively parallel computers. EBMT is a novel approach to machine translation that directly uses translation examples to carry out translation tasks. The authors have shown that EBMT attains substantially higher accuracy than traditional machine translation systems [Sumita and Iida, 1991]. However, EBMT is so computationally demanding that the system would run very slowly on sequential machines and even on a loosely-coupled multi-processor iPSC/2 [Intel, 1989].

The issue of processing speed is particularly critical for the success of the speech-to-speech translation system that we are striving to complete. In order to overcome the problem of computational cost, we have implemented EBMT on an IXM2 massively parallel associative memory processor [Higuchi et al., 1991] and a CM-2 connection machine [Thinking Machines Corp., 1989]. Experi-

mental results demonstrate that EBMT on the IXM2 exhibits the best performance and attains a response speed that will suffice for real-time tasks such as interpreting telephony.

The research reported in this paper is a reflection of two major research programs that are gaining increasing interest: the empirical approach to machine translation, and massively parallel artificial intelligence.

1.1 The Empirical Approach to Machine Translation

Since traditional MT systems rely upon the extensive use of rules, it is called the *rational approach*. However, traditional MT systems face the following difficulties: (1) improvement in translation quality is sluggish and (2) there is an intolerable increase in processing time, as system size increases.

A group of novel models has been proposed in recent years that make use of the increasing availability of a large-scale corpus and bulk processing power. The approach is called the *empirical approach* because it relies upon empirical knowledge such as the corpus and past translation pairs. There are several avenues in the *empirical approach*, however, including the example-based, or memory-based approach, the approach that we advocate, the statistical approach [Brown et al., 1988], and others. Some of the early works that depend solely upon the use of statistics, however, have begun to reveal their limitations such as the inability to assign an accurate probability for low frequency words.

EBMT has been proposed to overcome the difficulties that arise in traditional models and pure statistical models. The authors have shown that EBMT attains significantly higher accuracy over the traditional approach in the target word selection of function words, which is one of the most difficult tasks in translation by traditional MT [Sumita and Iida, 1991]. EBMT, however, is a computationally demanding approach because it requires retrieval of similar examples from a vast example database. Our initial implementation on a SparcStation-2 sequential workstation did not suffice for real-time translation. The iPSC/2, loosely-coupled multi-processor was some-

what better, but it did not achieve the level of response time required for speech-to-speech translation systems.

1.2 Massively Parallel Artificial Intelligence

Massively parallel artificial intelligence [Waltz, 1990; Kitano et al., 1991] is a novel subfield of AI made possible by the emergence of massively parallel computers. One of the mainstay approaches of massively parallel artificial intelligence is Memory-Based Reasoning (MBR) [Stanfill and Waltz, 1986]. MBR solves problems by assigning bulk data or rules directly to Processing Elements (PEs) of massively parallel machines, and carry out a data-parallel, similarity-based search to find the *best match* to a given problem. The virtue of this approach is that it exploits the inherent parallelism of the problems. Natural language processing is one of the domains suitable for massively parallel AI because many linguistic phenomena are individual in nature and cannot be handled by a small number of abstract rules.

The first research that advocated massively parallel natural language processing was *massively parallel memory-based parsing* [Kitano and Higuchi, 1991a; Kitano and Higuchi, 1991b]. The approach considers parsing to be pattern matching between an input sentence and a whole set of possible sequences of part-of-speech symbols. The specific implementation strategy involves parallel marker-passing which makes best use of the parallelism of massively parallel machines. Their experiment showed that parsing can be completed in a few milliseconds per sentence.

The research described in this paper provides independent evidence to support the validity of the approach for natural language processing. Specifically, this paper handles target word selection using EBMT. Target word selection is essential, and one of the hardest problems in machine translation. The central mechanism of word selection in EBMT is not the *exact match* but the *best match* based on semantic distance (See section 2.2). The *best match* improves the coverage of system knowledge.

The next section briefly introduces EBMT. Then, the problem of computational cost on a serial machine and on a multi-processor is described, followed by a description of the IXM2 massively parallel processor, on which we have implemented EBMT, and the performance results.

2 Example-Based Machine Translation

2.1 Rationale

In the early 1980s, Nagao proposed a novel model of machine translation called "Translation by Analogy," in order to conquer the bottleneck of traditional machine translation [Nagao, 1984]. Nagao assumes that a human being translates based on past translation experience. In other words, translations are created by mimicking similar translation examples. Since the end of the 1980s, many EBMT projects have begun to flourish in order to realize Nagao's model [Sato, 1991; Sato and Nagao, 1990; Sumita and Iida, 1991; Furuse and Iida, 1992; Kitano, 1991].

In EBMT, (1) a database that consists of translation

examples is prepared; (2) examples whose source part is most similar to the input phrase or sentence are retrieved from the example database; (3) the translation is obtained based on retrieved examples.

2.2 Semantic Distance

The above step (2) is to determine the set of examples whose semantic distance from the input is minimum. In other words, EBMT is the full retrieval of the most similar examples from the example array.

Suppose Input *I* and Example *E* are composed of *n* words. Our semantic distance measure [Sumita and Iida, 1991] 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 .

$$\sum_{k=1}^n d(I_k, E_k) \times w_k \quad (1)$$

The distance, $d(I_k, E_k)$ is determined based on a thesaurus. Each word corresponds to its concept in the thesaurus. The distance between words is reduced to the distance between concepts. 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, the distance, (k/n) is given to the classes in the *k*-th layer from the bottom ($0 \leq k \leq n$). The weight, w_k , is the degree to which the word influences the selection of the translation.

2.3 Advantages and Extensions

The authors [Sumita and Iida, 1991; Sumita and Iida, 1992] have demonstrated that EBMT's accuracy is superior to that of traditional MT in target word selection of function words (e.g., Japanese particles and English prepositions), moreover, EBMT has many advantages that traditional MT does not have.

The *best match* based on semantic distance is useful not only for target word selection but also for disambiguation in analysis and generation. Recently, TDMT [Furuse and Iida, 1992], a medium-sized¹ spoken language translation system fully utilizing the EBMT mechanism, has been implemented and is now being tested from various points of view. By now, promising prospects for translating not only phrases but also sentences and for disambiguating the syntactic structure, have been demonstrated.

3 The Problem of EBMT: Computational Cost

As mentioned in section 2.2, EBMT is basically full retrieval of similar examples from a large-scale example database. The processing time (*T*) is proportional to the number of examples (*N*) — $T = t \times N$, where *t* is the time to calculate the semantic distance between one example and the input.

In the actual system, EBMT is embedded as a sub-routine call, and is called many times during the translation of one sentence. Figure 3.1 shows the statistics

¹The vocabulary size is about 1,500.

on the number of EBMT calls required to translate one sentence. This is data from an experiment that translates 225 sentences by TDMT prototype, which makes the most use of EBMT to translate not only phrases but also sentences, as mentioned in section 2.3.

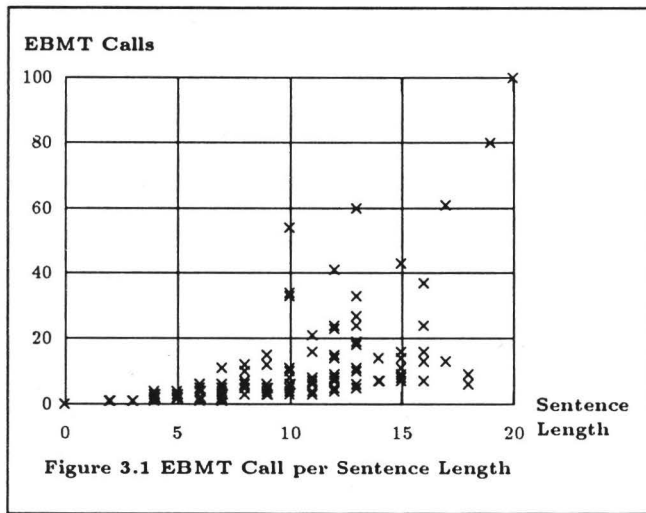


Figure 3.1 EBMT Call per Sentence Length

Figure 3.1 clearly illustrates that the number of EBMT calls rises according to sentence length. In the test corpus, the average sentence length is about 9.4 words and the average number of EBMT calls is about 9.5.

We have initially implemented EBMT on a SparcStation-2. The SparcStation-2 took 570 milliseconds for one EBMT call with 1,000 examples. Since one sentence requires an average of 9.5 EBMT calls, the sequential machine takes over several seconds just for EBMT calls. Moreover, this computing time increases in direct proportion to the number of examples. This performance is simply unacceptable for spoken language translation systems.

One of the obvious ways to overcome the problem of computing cost is the use of parallel computers. We have implemented EBMT on an iPSC/2 loosely-coupled multi-processor with 16 PEs. Examples are stored in local memory on each PE, and retrieval results are collected by message passing.

Figure 3.2 shows the response time with regards to the number of examples and the number of PEs. With 4,000 examples, the iPSC/2 (16PEs) took about 2 seconds for one EBMT call. Even with 1,000 examples, it took 570 milliseconds. Given the fact that on average 9.5 EBMT calls are made per sentence, the iPSC/2 requires several seconds for EBMT calls alone. Reconfiguring the iPSC/2 for 32, 64, or even 256 PEs would not improve the situation due to the significant communication overhead. In the iPSC/2, the latency for interprocessor communication is significant. In our application, in the 16-PE configuration, communication to retrieve examples took up 110 milliseconds out of 570 milliseconds. Since this communication overhead (C_p) increases as number of PEs increase ($C_p = c \times \log_2 p$: c is constant, p is number of PEs), a larger number of PEs results in a larger commu-

nication overhead. Based on our experiments, a 256-PE iPSC/2 would require about 220 milliseconds for communication whereas calculation of the semantic distance is completed in about 30 milliseconds (for 1,000 examples). Total computing time would be 250 milliseconds, which results in about 2.5 seconds for EBMT calls alone, which is far from satisfactory. Our conclusion is that the iPSC/2 loosely-coupled multi-processor does not suffice for our purpose.

Therefore, we decided to move our computing platform to massively parallel machines.

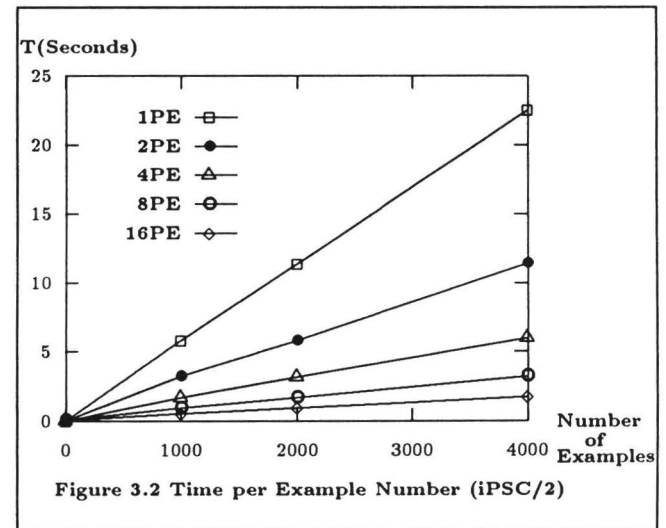


Figure 3.2 Time per Example Number (iPSC/2)

4 IXM2 Massively Parallel Associative Memory Processor

We have implemented EBMT on a massively parallel associative memory processor, the IXM2.

The IXM2 is the first massively parallel associative processor that clearly demonstrates the computing power of a large Associative Memory (AM). AM not only has a storage operation but also logical operations such as retrieving by content. *Parallel search* and *parallel write* operation are particularly important.

The IXM2 attains 256K-fold parallelism by using the NTT AM chip. Electrotechnical Laboratory designed the IXM2 for AI applications, particularly for semantic network processing. It was shown [Higuchi et al., 1991] that the IXM2 outperforms most other high performance machines.

The IXM2 consists of 64 processors, called *associative processors*, and 9 communication processors. The IXM2 is a co-processor under the control of a SparcStation. Each associative processor has an AM of 4K words of 40 bits, plus an IMS T800 Transputer(17.5 MHz).²

The decision to use AM chips driven by transputers, 32-bit CPUs, rather than thousands of 1-bit CPUs, is the major reason for the superior performance, processor

²The *associative processor* used in the experiment reported in this paper is the new version, with a T801 Transputer(25MHz) and faster SRAM.

efficiency, and cost performance of the IXM2 over other machines.

From the MT developers' point of view, we can build a highly scalable³ system that can improve the translation quality without degrading translation speed simply by adding examples to the AM chip.

5 Implementation

In the experiments, we translate Japanese noun phrases of the form "A の B" into English (A and B are nouns. "の" is an adnominal particle.) They are translated into various English noun phrases of the form such as "B of A," "B in A," "B for A," "B at A" and so on (A and B are English translations of A and B.) They are representative because the examples are composed of three⁴ variables (words) and they are difficult in traditional MT but EBMT can handle them well. The "A の B" examples were collected from the ATR corpus which consists of Japanese sentences and their English translations concerning a *conference registration* task.

5.1 Data Structure

5.1.1 Thesaurus

Each word has a three-digit decimal code of the concept to which the word corresponds. This clearly shows the positions of concepts in the thesaurus hierarchy.

5.1.2 Inputs and Examples

The data structure⁵ of the noun phrase, "京都の会議" [conference in KYOTO] is illustrated in Figure 5.1. "A の B" consists of the words of A and B (AW, BW), the code for A and B (AC, BC), and the word "の" (NO).⁶ The example database is stored in an array of structures in this form. The data are coded as an appropriate bit pattern on the AM of the IXM2.

AW	AC	NO	BW	BC
京都	357	の	会議	741
[word]	[code]	[word]	[word]	[code]

Figure 5.1 Data Structure for Input & Example

5.2 Algorithm

5.2.1 Distance Calculation

(a-1) **WORD Part** The distance is 0 or 1 depending on whether or not they match.

³In EBMT, translation quality is higher as the number of examples increases [Sumita and Iida, 1991].

⁴The typical number of variables in examples is two or three

⁵Here, for simplicity, the variables for marking in the AM, the variable, *distance* in the transputer's RAM, and the representation of polysemy(plural codes) are left out.

⁶Adnominal Particles do not have codes.

(a-2) **CODE Part** The distance explained in section 2.2 is directly calculated as shown in the table below without traversing the thesaurus hierarchy. Suppose the input code and example code are $CI = CI_1CI_2CI_3$, $CE = CE_1CE_2CE_3$.

Condition	Example	Distance
$CI_1CI_2CI_3 = CE_1CE_2CE_3$	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

5.2.2 Retrieval

(1) [WORD Exact Match]

Examples whose *AW*, *NO*, *BW* match the input are marked. If there is a match, then $step := 1$ ⁷, go to (4).

(2) [CODE Exact Match]

Examples whose *AC*, *NO*, *BC* match the input are marked. If there is a match, then $step := 2$, go to (4).

(3) [CODE Partial Match]

(3-1) Examples whose *NO* does not match the input are marked. The weight of *NO* is added to the variable *distance* of the examples.

(3-2) For each of *AC* and *BC*, the distance between codes is calculated according to (a-2), and distance $\times$ weight is added to the variable, *distance* of examples. $step := 3$

(4) [Example Collection]

If $step = 1$ or 2 , all marked examples are collected and the examples and the distance 0 are returned.

If $step = 3$, first, the minimum-distance is attained. Second, all the minimum-distance examples are collected and the examples and the minimum-distance are returned.

6 Experimental Results

Experiments have been carried out using 1,000 examples. The processing time is 270 milliseconds⁸ on the SparcStation-2 and 21 milliseconds on the IXM2 (See Figure 6.1).

The IXM2 attains a speed almost 13 times greater than that of the SparcStation-2. When a fully-equipped IXM2 (64 associative processors) is used, it would be able to load 64,000 examples. For example set of this size, the IXM2 would be able to attain a speed $13 \times 64 = 832$ times greater than that of the SparcStation-2.

The superior performance of the IXM2 can be attributed to the fact that the IXM2 carried out match processes (1), (2), and (3) using *parallel search* and *parallel write* for the AM. Distance calculations in step (3) are triggered in a constant time and executed in a linear time by the transputer. The example collection process (4) is executed linearly(See Figure 6.2).

⁷The variable *step* indicates the matched step.

⁸The performance figures hereafter are the average of above-mentioned three cases (1), (2) and (3). Thus, they are about half of the performance figures presented previously.

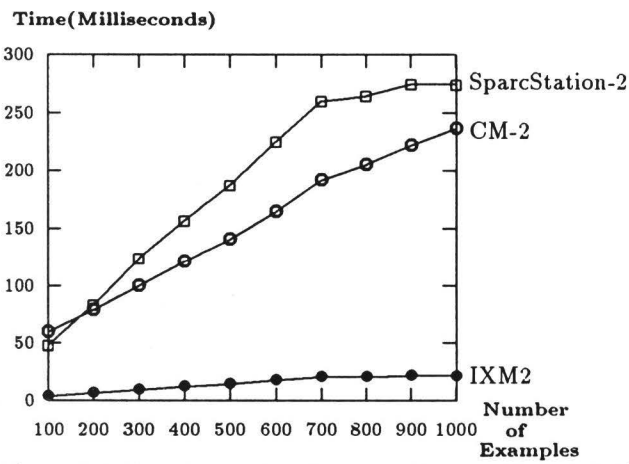


Figure 6.1 Time Comparison(SparcStation-2, CM-2, IXM2)

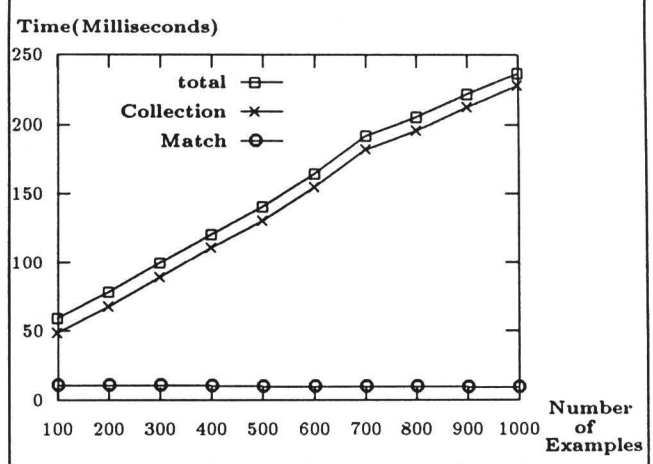


Figure 6.3 Match and Collection Time (CM-2)

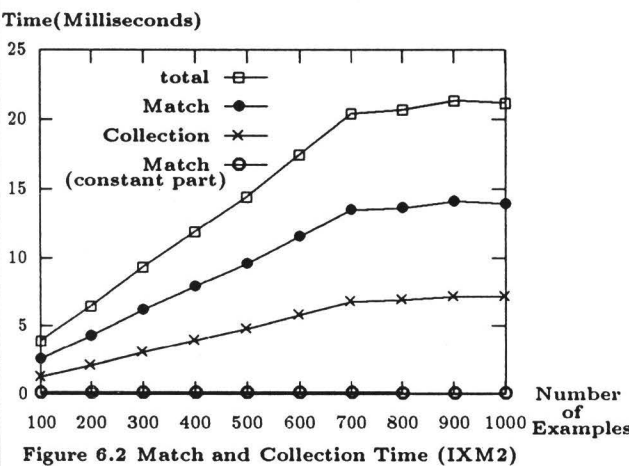


Figure 6.2 Match and Collection Time (IXM2)

Let us assume that we have N examples equally distributed on p PEs. This is a typical memory-processor relationship in array processors, such as the CM-2, and multi-processors, such as the iPSC/2. The processing time of each PE, which stores equally divided N/p examples, is $t \times N/p$ where t is the time required to calculate the semantic distance between one example and the input. The total processing time (T_p) is the sum of $t \times N/p$ and communication time (C_p), which is the time required for collecting the examples. The following equation gives the total processing time:

$$T_p = t \times \frac{N}{p} + C_p \quad (2)$$

The communication latency C_p is the major negative factor in many parallel processing applications. In the iPSC/2 with 16 PEs, EBMT results in $C_p = 110$ milliseconds using a standard communication algorithm. The C_p of a standard communication algorithm is as follows:

$$C_p = c \times \log_2 p \quad (3)$$

Therefore, if we keep increasing the number of PEs, the computing time at each PE can be minimized, but the communication overhead would be dominant. In the iPSC/2, 110 milliseconds are spent for communication between 16 PEs, and the communication overhead is expected to increase as the number of PEs increases. This creates a trade-off between improved computing time at each PE and communication overhead. This implies that there is a maximum performance barrier that iPSC/2-type architecture cannot overcome.

From discussions up to this point, we can derive the speedup rate (S):

$$S = \frac{p}{(1 + (c/t) \times (1/N) \times p \log_2 p)} \quad (4)$$

By measuring t and c , which are, respectively, roughly 7.4 milliseconds and 28 milliseconds for the iPSC/2, we can determine the performance barrier. $\frac{dS}{dp} = 0$ is the condition for finding the trade-off point.

On the other hand, however, one other massively parallel machine, the CM-2, took 240 milliseconds. Although CM-2 conducts match processes (1), (2) and (3) in a constant time, the example collection process (4) is executed linearly by the host machine, a SparcStation-2. Unfortunately, the communication between the CM-2 and the SparcStation-2 is done through the VME bus, which is very slow. Thus, the advantage of the data-parallel search was completely lost in the communication bottleneck of data collection by the host(See Figure 6.3).

7 Discussions

One of the interesting phenomena observed in this experiment is that of the trade-off between the number of processors and the communication overhead. The iPSC/2 and CM-2 largely depend upon interprocessor communication whereas the IXM2 minimizes the traffic by handing retrieval at each local transputer. The IXM2 still attains massive parallelism by using AM. This difference in design is decisive in our application.

As for the CM-2, the problem of the communication overhead is far more serious. The CM-2 took only 9 milliseconds to calculate each semantic distance. This is a quick process due to the data-parallel nature of the EBMT. This computing time is constant up to 64K examples because one processor has only one example to calculate. Despite this high performance at distance computing, the CM-2's overall response time is 240 milliseconds, indicating the seriousness of the communication overhead (See Figure 6.3).

The IXM2 architecture was superior over that of other machines because of the high performance example search by AM and the minimum communication overhead due to the use of transputers for example collection. Actually, the interprocessor communication required for the EBMT is the transmission of the chosen examples to the host and is negligible. Experimental data indicates that the IXM2 took 14 milliseconds for matching and 7 milliseconds for collection (See Figure 6.2).

8 Conclusion

The authors have demonstrated that EBMT on the massively parallel processor IXM2 attains a sufficiently fast response time to implement real-time speech-to-speech translation systems. While accuracy and incremental fine-tuning capability has been demonstrated in several studies reported so far, the problem of computing cost has been a major problem left unresolved.

We have implemented the EBMT system using substantial linguistic data, and on several types of machines such as the SparcStation-2, iPSC/2, CM-2, and IXM2.

Experimental results demonstrated that the IXM2 is the best computing platform for EBMT and justified our choice. The performance and the scaling capability of the IXM2 is desirable since a very large set of examples may be used in the near future to attain very high quality translation. Other machines such as the iPSC/2 and the CM-2 suffer from the trade-off between processor number and communication overhead.

From our experience with implementation in this project, we began to expect that massively parallel architecture is suitable not only for memory-based parsing and EBMT, but also for other natural language processing, such as n-gram based morphological analysis, which can be regarded as problems of retrieval from a large database. This would be another challenging topic to pursue.

In summary, we have confirmed the effectiveness of massively parallel artificial intelligence for EBMT. We expect our report to provide strong evidence to support the validity of massively parallel AI. The marriage of two new research fields — the empirical approach to machine translation and massively parallel artificial intelligence — is so far happy and successful.

Acknowledgements

Kitano is supported, in part, by the National Science Foundation's MIP-9009111 and the Pittsburgh Supercomputer Center's IRI-910002P.

References

- [Brown et al., 1988] Brown, P., Cocke, J., Della Pietra, S., Della Pietra, V., Jelinek, F., Mercer, R. and Roossin, P.: "A Statistical Approach to Language Translation," *Proc. of Coling '88*, pp.71-76, (1988).
- [Furuse and Iida, 1992] Furuse, O. and Iida, H.: "Cooperation Between Transfer and Analysis in Example-Based Framework," *Proc. of Coling '92*, pp.645-651, (1992).
- [Higuchi et al., 1991] Higuchi, T., Kitano, H., Handa, K., Furuya, T., Takahashi, N. and Kokubu, A.: "IXM2: A Parallel Associative Processor for Knowledge Processing," *Proc. of AAAI-91*, Anaheim(1991).
- [Intel, 1989] Intel Scientific Computers, *iPSC/2 User's Guide*, 1989.
- [Kitano, 1991] Kitano, H.: "ΦDM-Dialog: An Experimental Speech-to-Speech Dialog Translation System," *IEEE Computer*, June, pp.36-50, (1991).
- [Kitano and Higuchi, 1991a] Kitano, H. and Higuchi, T.: "Massively Parallel Memory-Based Parsing," *Proc. of IJCAI-91*, pp.918-924, (1991).
- [Kitano and Higuchi, 1991b] Kitano, H. and Higuchi, T.: "High Performance Memory-Based Translation on IXM2 Massively Parallel Associative Memory Processor," *Proc. of AAAI-91*, pp.149-154, (1991).
- [Kitano et al., 1991] Kitano, H., Hendler, J., Moldovan, D., Higuchi, T. and Waltz, D.: "Massively Parallel Artificial Intelligence," *Proc. of IJCAI-91*, pp.557-562, (1991).
- [Nagao, 1984] Nagao, M.: "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, pp.173-180, (1984).
- [Sato, 1991] Sato, S.: "Example-Based Machine Translation," *Doctorial Thesis*, Kyoto University, (1991).
- [Sato and Nagao, 1990] Sato, S. and Nagao M.: "Toward Memory-Based Translation," *Proc. of Coling-90*, Helsinki, pp.247-252, (1990).
- [Stanfill and Waltz, 1986] Stanfill, C. and Waltz, D.: "Toward Memory-Based Reasoning," *CACM*, Vol. 29, No. 12, pp.1213-1228, (Dec. 1986).
- [Sumita and Iida, 1991] Sumita, E. and Iida, H.: "EXPERIMENTS AND PROSPECTS OF EXAMPLE-BASED MACHINE TRANSLATION," *Proc. of 29th ACL*, pp.185-192, (June 1991).
- [Sumita and Iida, 1992] Sumita, E. and Iida, H.: "Example-Based NLP Techniques -A Case Study of Machine Translation-", *Statistically-Based NLP Techniques Workshop of the AAAI-92*, pp.90-97, (1992).
- [Thinking Machines Corp., 1989] Thinking Machines Corp.: "Model CM-2 Technical Summary," *Technical Report TR89-1*, (1989).
- [Waltz, 1990] Waltz, D.: "Massively Parallel AI," *Proc. of AAAI-90*, (1990).