Massively Parallel Memory-Based Parsing*

Hiroaki Kitano[1,2] and Tetsuya Higuchi[3]

Center for Machine Translation[1]
Carnegie Mellon University
Pittsburgh, PA 15213 U.S.A.
hiroaki@cs.cmu.edu

NEC Corporation[2]
5-33-1 Shiba, Minato-ku
Tokyo 108, Japan

Electrotechnical Laboratory[3]
1-1-4 Umezono, Tsukuba
Ibaraki 305 Japan
higuchi@etl.go.jp

Abstract

This paper discusses a radically new scheme of natural language processing called *massively parallel memory-based parsing*. Most parsing schemes are rule-based or principle-based which involves extensive serial rule application. Thus, it is a time consuming task which requires a few seconds or even a few minutes to complete the parsing of one sentence. Also, the degree of parallelism attained by mapping such a scheme on parallel computers is at most medium, so that the existing scheme can not take advantage of massively parallel computing. The massively parallel memory-based parsing takes a radical departure from the traditional view. It views parsing as a memory-intensive process which can be sped up by massively parallel computing. Although we know of some studies in this direction, we have seen no report regarding implementation strategies on actual massively parallel machines, on performance, or on practicality accessment based on actual data. Thus, this paper focuses on discussion of the feasibility and problems of the approach based on actual massively parallel implementation using real data. The degree of parallelism attained in our model reaches a few thousands, and the performance of a few milliseconds per sentence has been accomplished. In addition, parsing time grows only linearly (or sublinearly) to the length of the input sentences. The experimental results show the approach is promising for real-time parsing and bulk text processing.

1. Introduction

This paper presents a radically new scheme of natural language processing called *massively parallel memory-based parsing*. We will report the experimental results of our scheme actually implemented on massively parallel machines, and discuss the benefits and problems of the approach. Specifically, we will examine its performance and memory requirements — we will show that parsing can be completed in a few milliseconds, and the memory requirement is within practical limits.

Massively parallel memory-based parsing was inspired from ideas of the memory-based reasoning and case-based reasoning which place memory as the basis of reasoning. These paradigms are, by definition, memory-intensive and, usually assume the use of massivelly parallel machines for the implementation of practical systems. So far, some successful results on massively parallel implementation of memory-based reasoning systems have been published in such areas as word pronunciation [Stanfill, 1988] and classification of census data [Waltz, 1990]; surprisingly, no report has been made on the application of the idea to natural language parsing with full syntactic and semantic analysis. The only studies we have today in this direction are the Direct Memory Access Parser (DMAP) [Riesbeck and Martin, 1986], DMTRANS machine translation system [Tomabechi, 1987] [Kitano et. al., 1989], and the ΦDMDIALOG speech-to-speech translation system [Kitano, 1990]. However, DMAP and DMTRANS are only implemented on very small scale (less than 100 words) simulated on serial machines, leaving the scalability and practicality open to question. ΦDMDIALOG is larger (more than 500 words) and more sophisticated system, yet it was only recently that versions of the system was implemented on massively parallel machines [Kitano and Higuchi, 1991] [Kitano, et. al., 1991]. Also, we have not seen any detailed discussions on the memory-based or case-based parsing regarding implementation strategies and theoretical claims based on actual data. In this paper, we report the memory-based parsing model actually implemented on massively parallel computers, and discuss the viability of the approach using actual data.

We use three corpora of spoken utterances. These are: (1) ATR (ATR Interpreting Telephony Research Laboratories) conference registration task which contains 329 utterances and the vocabulary size of 450 extracted from simulated telephone conversations. (2) The DARPA (Defense Advanced Research Project Agency) resource management task contains 3345 sentences and the vocaburaly size of 997. (3) The CNN Prime News consists of 453 sentences. The CNN Corpus is the real-world data from actual cable broadcasting. Some of the sentences are very long, but can be segmented into two or three independent sentences (segmented data is referred as 'CNN Segmented', and the raw data is simply refered as 'CNN Prime News').

2. Memory-Based Parsing

Memory-Based Parsing was inspired by the memory-based reasoning paradigm proposed in [Stanfill and Waltz, 1988] and [Stanfill and Waltz, 1986]. The basic idea of memory-based reasoning places memory at the foundation of intelligence. It assumes that large numbers of specific events are stored in memory, and response to new events is handled by first recalling past events which are similar to the new input, and invoking actions associated with these retrieved

*This work is supported in part by the Pittsburgh Supercomputing Center under grant TRA900105P and IRI-910002P.

events to handle the new input[1]. This idea runs counter to most AI approaches which place rules or hueristics as the central thrust of reasoning. For example, traditional parsing has been considered a rule-based or principle-based process which comprises serial rule application procedures. However, the massively parallel memory-based parsing makes a radical departure from the traditional view. It considers parsing to be a memory-intensive process which can be sped up by massively parallel computing. In the memory-based parsing model, parsing is viewed as a memory-search process which locates the past occurrence of similar sentences, and the interpretation is built by activating the past occurrence. We believe that *memory and their associations plays the central role in many intelligence tasks.* This view is similar to the idea of Direct Memory Access Parsing [Riesbeck and Martin, 1986], except that our model does not contain adaptation and refinement processeses required in case-based reasoning, and there is no consideration of a massively parallel implementation in the DMAP model as it presently exists. Actually, the main focus of the DMAP is case-adaptation for parsing, and it does not pay much attention to parsing with large instances of sentences. However, the basic parsing process is quite similar, so by testing our model we can also examine the practicality of the DMAP model for real-world tasks.

The first issue is performance. Traditional parsing is a time consuming task. A few seconds or even a few minutes is required to complete the parsing of one sentence. The most efficient parsing algorithm known to date is Tomita's generalized LR parser which takes less than $O(n^3)$ for most practical cases [Tomita, 1986]. In addition, the Earley type algorithms degrade its performance as size of grammar increases ($O(G^2)$). Furthermore, extensive use of unification, which is a computationally expensive operation, in recent grammar theories substantially undermines the speed of processing. Efforts to parallelize these traditional approachs have only a limited contribution to attain turely real-time parsing system. Reasons for this include: (1) the level of parallelism attained by implementing a parallel version of the traditional parsing scheme is rather low, (2) serial application of piecewise rules causes combinatorial explosion which leads to substantial performance degradation as input length gets longer and/or as size of grammar grow larger, and (3) unification is essentially a sequential operation which does not gain much from parallelization.

In our model, serial rule application is eliminated and replaced by a parallel memory-search process, Also, syntactic structure and interpretation are pre-indexed so that expensive unification operation is no longer necessary, or sufficiently minimized. This approach has not been taken in the serial machine because the approach will face a trade-off between improved efficiency due to pre-expansion and degradation due to an increase in search cost. Since each instance of memory activation can be processed independently on a massively parallel machine, we expect that the level of parallelism will exceed at least 1,000 which will lead to a dramatic increase in performance.

[1]One of the major differences of our model from other memory-based reasoning models is that we do not use similarity-based memory matching. This is due to the lack of sound domain theory of similarity matching in parsing. A similarity-based matching is only possible when the syntactic pattern of the input sentence is very close (we do not have measurement of closeness itself) to one of the instances. However, if a syntactic structure is different, we can not make use of past instances.

Other than the performance aspect, there are several problems which need to be examined in order to claim that memory-based parsing is a viable model. Perhaps the most significant issue is the memory requirement. Since the memory-based reasoning paradigm requires an extensive number of past instances to solve new problems, the critical issue is whether the number of necessary cases converges with a practical number in the given task domain. In natural language processing, the productivity of language dictates that human beings are capable of producing an infinite number of sentences. Thus, if we store all the sentences which we could possibly encounter, we need either an infinite memory space, or finite but astronomical space. Obviously, such a naive approach should be rejected. The approach we do take is to use abstract instances of sentences. For example, instead of storing sequences of words, we store sequences of syntactic categories such as *[det n v p n]* or semantic grammar templates such as *[agent want-to attend event]*. By using abstract templates, the memory space required to cover the task domain will be significantly reduced [2]. Although theoretically, an infinite number of syntactic patterns need to be stored to process all possible sentences, a finite number of syntactic patterns can cover a fairly significant percentage, say 99.9%, of possible input sentences when the length of the sentence has a specific upper-boundary[3]. Still, it is open to question whether or not the necessary number of instances converges within a practical size. Observation of convergence of the coverage by a finite number of syntactic patterns is one of the major purposes of the experiments in this paper.

3. Experimental Implementation

This section describes the implementation used in the experiments in this paper. It should be understood that the idea of memory-based parsing is new and that it is in the early stages of development. Thus the specific implementation described here should be regarded as an example of implementation, not the definitive implementation of the memory-based parser. In fact, we will discuss some enhancements later. The experimental implementation has two major parts: a massively parallel associative processor IXM2 and a memory-based parser implemented on the IXM2.

3.1. The Massively Parallel Associative Processor IXM2

IXM2 is a massively parallel associative processor designed and developed at the Electrotechnical Laboratory [Higuchi et. al., 1991]. It is dedicated to semantic network processing using marker-passing.

IXM2 consists of 64 processors, called *associative processors*, which operate with associative memory, each of which has a memory capacity of 256K words by 40 bits. Each associative processor is connected to other associative processors through network processors. The structure of the IXM2 is shown in figure 1.

An associative processor consists of an IMS T800 transputer, 8 associative memory chips, RAM, link adapters, and

[2]An alternative approach is to store a large set of sentences in its surface sequence, and use similarity-based matching to cover unknown inputs. See [Sumita and Iida, 1991] for such an approach.

[3]In the section 5, we will demonstrate that we approximately limit the maximum length of sentences in the spoken dialogues.

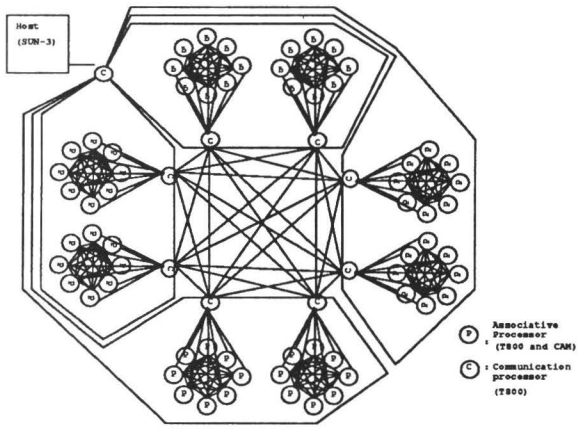

Figure 1: Structure of the IXM2 Associative Memory Processor

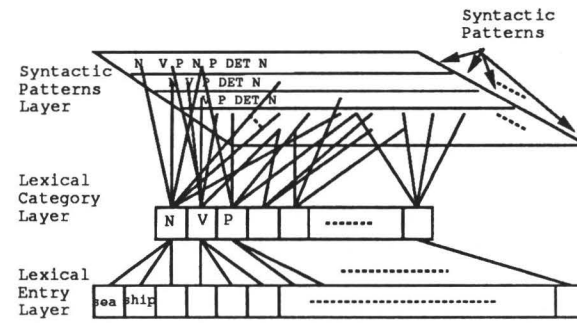

Figure 2: Overall Architecture of the Syntactic Recognition Part

001	N	V-BSE	DET	N	
002	N	V-BSE	N		
003	N	BE-V	V-PAS	PP-by	N

Table 1: A Part of Pre-Expanded Syntactic Structures (Simplified)

associated logic. When operated at 20 MHz clock, T800 attains 10 MIPS [Inmos, 1987]. Each associative memory chip is a 20 Kbit CAM (512 words × 40 bits) manufactured by NTT. The IXM2 has 64 such processors, thus attaining 256K parallelism which is far larger than 64K parallel of the Connection Machine [Hillis, 1985]. This high level of parallelism allows us to implement practical memory-based systems. The design decision to use associate memory chips driven by 32 bit CPUs, instead of having thousands of 1-bit CPUs, is the major contributing factor for performance, processor efficiency, and cost performance.

3.2. Organization and Algorithm of the Parser

We describe the organization and algorithm of the memory-based parser on the IXM2. As an experimental implementation designed to test the practicality of the approach, we employed a flat memory structure, i.e. no hierarchy was used to encode syntactic patterns. This is because the flat structure is the most memory-intensive way of implementing the memory-based parsing model. Thus, should this implementation be judged to be practically useful, other versions which use a more memory-efficient implementation can also be judged to be practical.

The system consists of two parts: a syntactic recognition part on the IXM2 and a semantic interpretation part on the host computer.

For the syntactic recognition part on the IXM2, the overall architecture is shown in figure 2. The memory consists of three layers: a lexical entry layer, a syntactic category layer, and a syntactic pattern layer.

Lexical Entry Layer: The lexical entry layer is a set of nodes each of which represents a specific lexical entry. Most of the information is encoded in lexical entries in accordance with modern linguistic theories such as HPSG[Pollard and Sag, 1987], and the information is represented as a feature structure. It is a straightforward task to represent huge numbers of lexical entries on the IXM2.

Syntactic Category Layer: The second layer comprises a group of nodes representing the syntactic features. Perhaps the most important feature for parsing is the head major category, generally known as the syntactic category. In the specific implementation examined in this paper, we use the head major category as a single feature to index syntactic structures. However, it is also possible to incorporate other features to index syntactic structures. The choice of features to be incorporated largely depends on the design decision on how much the constraint checks to be conducted on each processor or on the host computer.

Syntactic Patterns Layer: All possible syntactic structures are directly mapped onto the associative memory as a syntactic patterns layer. As mentioned earler, the syntactic structure is a flat sequence of syntactic categories which can be generated from the given grammar or from a corpus of training sentences. Table 1 shows a part of simple syntactic structure loaded on the associative memory. Grammatical constraints can be incorporated when expanding grammar rules. It allows for a recursive structure so that the number of actual syntactic structures loaded is less than the actual number of syntactic patterns the system can accept.

The degree of constraints which are incorporated in the expanded syntactic structures largely affects the memory requirements and the processing load on the host processor. If only the head major category is incorporated, most constraint checks must be done by the host computer or at the transputer. On the other hand, if all constraints are incorporated in expanding grammar, the number of possible syntactic structures will be explosive and it will require far more associative memory chips. In this experiment, we only used the head major category (such as NOUN, VERB), thus most constraint processing is done at each transputer and at the host processor. It is also possible to use more subdivided symbols at the cost of memory requirements.

In the host computer (SUN-3/250), the case-role binding table is pre-compiled which indicates correspondence between case-roles and word positions. Table 2 shows a part of a simple case-role binding table. Each position in the table is associated with actions to be taken in order to build meaning representation. In building the meaning representation, the program resides on the host computer and carries out

001	ACTOR	ACTION	DET	OBJECT	
002	ACTOR	ACTION	OBJECT		
003	OBJECT	was	ACTION	by	ACTOR
Words	John	was	kicked	by	Mary

Table 2: Case-Role Table (Simplified)

role-bindings and some constraint checks depending on how the constraints are incorporated into the syntactic recognition part. If there are ambiguous parses, more than two items in the table need to be processed. However, it should be noted that all items which are notified from the IXM2 are already known to be accepted parsing hypotheses as far as syntactic structure is concerned. This architecture drastically minimizes the number of operations required for parsing by eliminating operations on parses which turn out to be false.

The algorithm is simple. Two markers, activation markers (A-Markers) and prediction markers (P-Markers) are used to control the parsing process. A-Markers are propagated through the memory network from the lexical items which are activated by the input. P-Markers are used to mark the next possible elements to be activated. A general algorithm follows:

1. Place P-Markers at all first elements of the syntactic patterns.
2. Activate the lexical entry.
3. Pass the A-Marker from the lexical entry to the Syntactic Category Node (SCN).
4. Pass the A-Marker from the SCN to the elements in the Syntactic Patterns.
5. If the A-Marker and a P-Marker co-exist at an element in the Syntactic Pattern,
   then the P-Marker is moved to the next element of the Syntactic Pattern.
6. If there are no more elements, the syntactic pattern is temporarily accepted,
   and a pattern ID is sent to the host or local processors for semantic interpretation.
7. Repeat 2 thru 6, until the end of the sentence.

On the host computer or on the 64 T800 transputers, the semantic interpretation is performed for each hypothesis. The general flow follows:

1. Receive the syntactic pattern ID from the syntactic recognition part.
2. If words remain in the sentence, then ignore the ID received.
3. If no words remain, perform semantic interpretation by executing the functions associated with each hypothesis in the table. Most operations are reduced to a bit-marker constraint check and case-role bindings at compile time.

4. Performance

We carried out several experiments to measure the system's performance. Figure 3 shows the syntactic recognition time against sentences of various lengths. Syntactic recognition

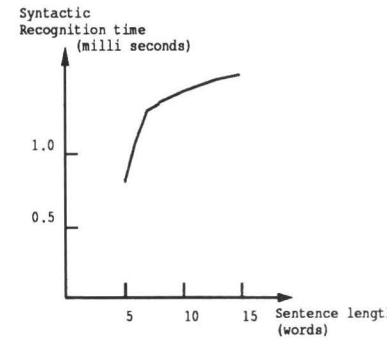

Figure 3: Syntactic Recognition Time vs. Sentence Length

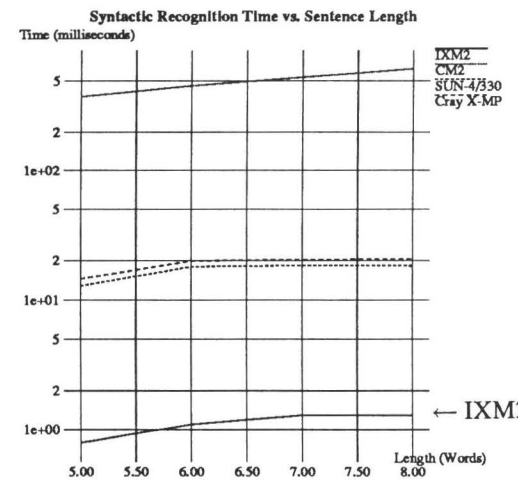

Figure 4: Comparison of Syntactic Recognition Time

at milliseconds order is attained. This experiment uses a memory containing 1,800 syntactic patterns. On average, 30 syntactic patterns are loaded into each associative processor. Processing speed improves as parsing progresses. This is because the computational costs for a sequential part in the process is reduced as number of hypotheses activated decreases. There is one sequential process which checks active hypothese on each 64 transputer. During this process, the parallelism of the total system is 64. Discussion of processor loading factor will be given in section 6.3.

It should be noted that this speed has been attained by extensive use of associative memory in the IXM2 architecture – simple use of 64 parallel processors will not attain this speed. In order to illustrate this point, we measured the performance of the SUN-4/330, CM-2 Connection Machine, and Cray X-MP with only 30 syntactic patterns which is equivalent to a single processor of the IXM2. The program on each machine uses an optimized code for this task in C language. The experimental results are drawn on figure 4. The IXM2 is almost 16 times faster than that of the SUN-4/330 and Cray X-MP even with such a small task[4] The CM-2 Connection Machine is very slow due to a communication bottleneck between processors. While both the IXM2 and the SUN-

[4]Cray X-MP is very slow in this experiment mainly due to its sub-routine call overhead. We have tested this benchmark on a Cary X-MP in Japan and at the Pittsburgh SuperComputing Center, and obtained the same result. Thus this is not hardware trouble or other irregular problem.

No. of Patterns	IXM2	CM2	SUN-4	Cray
10	0.7	608.4	4.4	4.7
30	1.3	620.8	18.2	20.4

Table 3: Syntactic Recognition Time vs. Grammar Size (milliseconds)

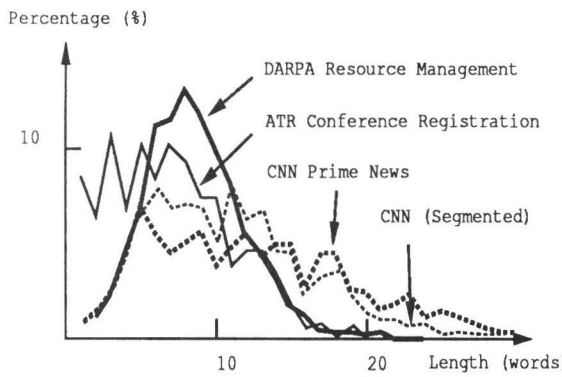

Figure 5: Distribution by Sentence Length

4/330 use a CPU of comparable speed, the superiority of the IXM2 can be attributed to its intensive use of the associative memory which attains a massively parallel search.

Next we examine the scaling property of both systems. Figure 3 shows the performance for a sentence of length 8, for syntactic patterns of size 10 and 30. While a single processor of the IXM2 maintains less-than-linear degradation, the SUN-4/330 and Cray X-MP degrades more than linearly. It should be noted that 30 syntactic patterns in other machines literally means 30 patterns, but in the single processor in the IXM2, it means 1,800 patterns when all 64 processors are used.

It is expected that the larger task set would demonstrate a dramatic difference in total computation time. The IXM2 can load more than 20,000 syntactic patterns which is sufficient to cover the large vocabulary tasks currently available for speech recognition systems. With up-to-date associative memory chips, the number of syntactic patterns which can be loaded on the IXM2 exceeds 100,000. Also, extending the IXM2 architecture to load over one million syntactic patterns is both economically and technically feasible.

5. Memory Requirements

While the high performance of memory-based parsing on a massively parallel machine has been clearly demonstrated, now we look into its memory requirement. It is well acknowledged that the productivity of language dictates that we can produce an infinite number of sentences. Even if we consider only vaild syntactic patterns, it will be infinite when no restriction has been applied. Obviously, we can not create and encode an infinite number of syntactic patterns. To advocate the memory-based parsing, we need to demonstrate that, *in practice*, the number of syntactic patterns actually used is finite, or it can be approximated by a finite number of syntactic patterns.

It should be noted that an infinite number of sentences can be produced when any of three assumptions stands:

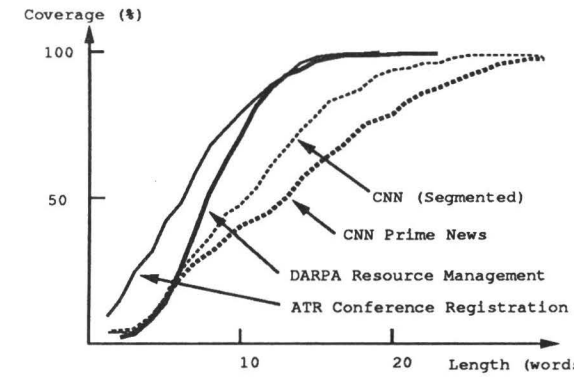

Figure 6: Coverage by Sentence Length

Assumption 1: Infinite Vocabulary

Assumption 2: Infinite Grammar Rules

Assumption 3: Infinite Sentence Length

In the other words, (1) finite vocabulary, (2) finite grammar rules, and (3) finite sentence length are the conditions for finite productivity of language. However, assumption 1 and 2 can be ignored since most of parsers only have finite vocabulary and grammar at run time. Also, people do not acquire infinite vocabulary and grammar rules at a point in the dialogue. Increase in vocabulary and grammar are more long term effects. What makes our model different from the traditional models depends upon whether the third assumption stands or not. If the third assumption is false, our model is, at least, equivalent to traditional models in its sentence productivity. Therefore, we counted the number of sentences of each length. Surprisingly, the ATR and DARPA corpora show very similar characteristics (figure 5): both have a peak of sentences between 7 to 9 words long. CNN has longer sentences, though it too has peaks in 7 to 13 words length. The ATR sentences' maximum length was 19 words and that of the DARPA corpus was 24. CNN Prime News was 48, and CNN segmented was 35. 99.7% of sentences from ATR, DARPA, and CNN Segmented corpus are less than 25 words length. From this data, we can expect that most sentences are within manageable length. Thus, we conclude that *in practice* sentence which can be produced at a given time point is finite.

Next we examine that if, in practice, the number of syntactic structures which appear in the given task domain will saturate at a certain number. Empirical observation using a corpus taken from the DARPA task shows that it does converge when it is in a restricted domain (figure 7). However, the number of syntactic patterns necessary to cover the task domain was 1,500 with the flat structure, and it was reduced to 900 with a simple hierarchical network. Since IXM2 is capable of loading over 20,000 syntactic patterns, the model is capable of covering the task even with the flat memory approach, and much wider domain can be covered with hierarchical model. However, a larger scale of experiment will be necessary to see if the number of syntactic patterns saturates, and where it saturates. We are currently investigating this issue using a large corpus from real world data such as CNN.

Independently, we have carried out an experiment to cover a given domain based on syntactic patterns pre-expanded from a set of grammar rules. We pre-expanded syntactic patterns from a set of context-free rules to see the memory requirements. A set of 6 basic grammar rules will produce about 2,000 patterns when the maximum length is 10 words, and

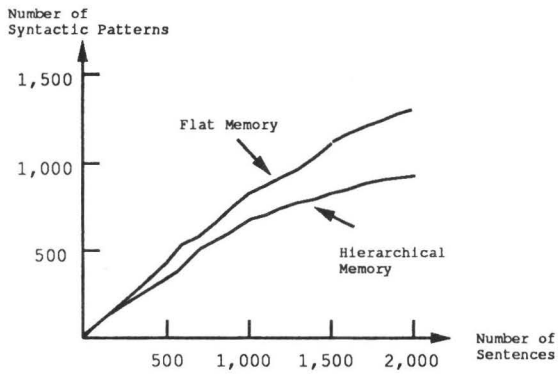

Figure 7: Training Sentences vs. Syntactic Patterns

about 20,000 patterns when the maximum length is 15 words. However, this has been reduced to 1/20 by using local networks which handle noun-noun modifications, adjective-noun modifications, etc. Thus, by imposing additional constraints, pre-expansion of syntactic patterns from a set of grammar rules is also feasible, and can be loaded on IXM2. In addition, it should be noted that not all syntactic patterns are actually used in the real world, thus the number of syntactic patterns that we really need to load on the machine would be far smaller. Psycholinguistic study shows that there is an upper-bound in the complexity of sentences which people can process [Gibson, 1990]. The hypothesis that the number of syntactic patterns that actually appears in the given task is relatively small can be independently confirmed. Nagao [Nagao, 1989] reported that syntactic patterns appeared in the title of over 10,000 scientific papers were around 1,000, and it was reduced to just 18 with simple reduction rules. While we can only confirm our hypothesis on the basis of our experiments on the small and medium size domains, increasing availability of large memory space and large number of processors provided by massively parallel machines offers a realistic opportunity that massively parallel memory-based parsing can be deployed practical tasks.

6. Discussions

6.1. Enhancement I: Hierarchical Memory Network

Use of the hierarchical memory network model reduces memory requirements by layering the levels of abstractions incorporated in the memory, but at the cost of building such networks which is not a trivial task. Figure 7 shows an example of the memory saving effect of the hierarchical memory network. By incorporating levels of abstractions (such as surface sequences, generalized cases, and syntactic rules) the memory requirements can be reduced significantly without undermining the benefits of the memory-based approach. They also exemplify an extended notion of the *phrasal lexicon*[Becker, 1975]. Actually, this model has been implemented in the ΦDMDIALOG speech-to-speech dialogue translation system, and has been proven to be useful for spoken language understanding systems. We have implemented a memory-based translation system using hierarchical memory network, and attained the milliseconds performance [Kitano and Higuchi, 1991]. Also, [Kitano, et. al., 1991] reports the similar idea has been implemented on the Semantic Network Array Processor (SNAP) [Moldovan et. al., 1990], and obtained a comparable performance. It should be cautioned, however, that excessive

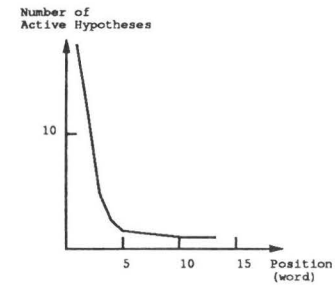

Figure 8: Number of Active Hypotheses per Processor

abstraction and hierarchical networks would undermine performance and quality of translation, because it would be close to rule-based systems. Use of abstraction and hierarchical networks should be minimized and should be used only when sufficient memory space was not allocated to cover a wide range of input sentences.

6.2. Enhancement II: Linguistic Knowledge

While we are currently using a relatively simple syntactic constraint and semantic interpretation mechanism using the case-role table, there is room for improvements. First, the syntactic constraints can be compiled as a large finite-state network so that some additional mechanisms using marker-passing can effectively incorporate syntactic constraints. Second, instead of using the case-role table, we can load functional descriptions and partially build f-structures so that only lexical insertion will be performed to complete final f-structure (Levin and Gates; personal communication). Most heavy operations such as unification are performed in compile time to create a partial f-structure, and only a minimal operation is left to run time. With this approach, the unification operation will be minimized, if not eliminated, for the number of words in the sentence.

6.3. Hardware Architecture for Memory-Based Parsing

First, the IXM2-type associative memory machine has the advantage in cost performance. In the IXM2, the associative memory stores syntactic patterns, and it allows for a massively parallel search (the current implementation allows for a parallel search up to 256K), while limiting the number of processors to 64.

Figure 8 shows the number of active hypotheses per one associative processor. Although it starts with a high processing load where a significant percentage of hypotheses are activated, the number of hypotheses decreases drastically as the processing proceeds. Since the IXM2 uses associative memory chips to store syntactic patterns, no processor will be idle unless all the hypotheses assigned to the processor are eliminated. However, in other massively parallel machines that assign processors to all the hypotheses, most of the processors will be idle because most of the hypotheses will be eliminated as the processing progresses. Since the use of associative memory chips would be far cheaper than processor chips to store and carry out the operations necessary in the implementation in this paper, the IXM2's architecture would be more cost effective than other architectures for this task.

Although there are cases that performance and functionalities benefit from assigning one processor for one hypothesis, (such cases involve complex calculations of probabilistic

measures, a source of activation check, and dynamic reconfiguration of the network) the associative memory-based architecture as seen in the IXM2 suffices for many memory-based reasoning tasks. In most memory-based reasoning tasks, similarity matching uses relatively simple similarity measures from numeric computations which can be computed on associative memory. Even in cases which require complex computations, the IXM2 is expected to maintain high performance with better cost effectiveness, because a sixty four 32 bit CPU can distributively perform higher-levels of symbolic and numeric operations. Thus, the IXM2's architecture which we advocate in this paper is a cost effective architecture not only for the memory-based parser, but also for more general memory-based reasoning systems.

7. Conclusion

In this paper, we proposed a *massively parallel memory-based parsing*. We have shown, using data obtained from our experiments, that the *massively parallel memory-based parsing* is a promising approach to implement a high-performance real-time parsing system for certain task domains. Major claims and observations made by our experiments are:

- The massively parallel memory-based parsing attains real-time parsing when implemented on a massively parallel machine. Our experiments using the IXM2 associative memory processor shows that syntactic recognition is completed in less than 2 milli-seconds. The system not only attains milli-second order parsing performance, but also exhibits a desirable scaling property. The parsing time grows only linearly (or sublinearly) to the size of the inputs ($\leq o(n)$). Also, the parsing time required grows only sublinearly to the number of syntactic patterns loaded. This scaling property is the real benefit of using a massively parallel machine.
- Massively parallel memory-based parsing, even in its simplest form, can be implemented within practical memory and processor requirements, when designed for suitable task domains. Our observation from spoken language corpora demonstrates that the length of sentences converges within a managable length, and the number of syntactic patterns also converges into a practical scale. This enables us to implement the memory-based parsing model using massively parallel machines that already exist. With the possible development of larger scale massively parallel machines such as the one targeted by DARPA for TeraOps by 1995 [Waltz, 1990], the possibility of the large scale massively parallel memory based parsing would be within sight.
- The IXM2's architecture is cost effective for memory-based reasoning tasks. The use of associative memory to attain a high level of parallelism (256K, in the current implementation) allows us to build a practical memory-based system with possibly lower resource requirements. While only a few parts of memory are actually involved in solving a specific problem, most processors could be idled when a full fine-grained processor architecture is used. The IXM2 is one of the ideal and cost effective architectures for building practical and large-scale memory-based systems.

Acknowledgements

We would like to thank the members of the Center for Machine Translation at Carnegie Mellon University, particularly Jaime Carbonell and Masaru Tomita, and the members of Electrotechnical Laboratory for discussions and support. Also, we would like to thank Hitoshi Iida and Akira Kurematsu at the ATR Interpreting Telephony Research Laboratories for allowing us to use their corpus, and for their continuous encouragement.

References

[Becker, 1975] Becker, J. D., *The Phrasal Lexicon*, BBN, Technical Report, 3081, 1975.

[Gibson, 1990] Gibson, E., "Memory Capacity and Sentence Processing," *Proceedings of ACL-90*, 1990.

[Higuchi et. al., 1991] Higuchi, T., Kitano, H., Furuya, T., Kusumoto, H., Handa, K., and Kokubu, A., "IXM2: A Parallel Associate Processor for Knowledge Processing," *Proceedings of the National Conference on Artificial Intelligence (AAAI-91)*, 1991.

[Hillis, 1985] Hillis, D., *The Connection Machine*, The M.I.T. Press, 1985.

[Inmos, 1987] Inmos, *IMS T800 Transputer*, 1987.

[Kitano, et. al., 1991] Kitano, H., Moldovan, D., Um, I., Cha, S., "High Performance Natural Language Processing on Semantic Network Array Processor," *Proceeding of the International Joint Conference on Artificial Intelligence (IJCAI-91)*, 1991.

[Kitano and Higuchi, 1991] Kitano, H. and Higuchi, T., "High Performance Memory-Based Translation on IXM2 Massively Parallel Associative Memory Processor," *Proceeding of the National Conference on Artificial Intelligence (AAAI-91)*, 1991.

[Kitano, 1990] Kitano, H., "ΦDMDIALOG: A Speech-to-Speech Dialogue Translation System," *Machine Translation*, 5, 1990.

[Kitano et. al., 1989] Kitano, H., Tomabechi, H., and Levin, L., "Ambiguity Resolution in DMTRANS PLUS," *Proceedings of the European Chapter of the ACL*, 1989.

[Moldovan et. al., 1990] Moldovan, D., Lee, W., and Lin, C., *SNAP: A Marker-Passing Architecture for Knowledge Processing*, Technical Report PKPL 90-4, Department of Electrical Engineering Systems, University of Southern California, 1990.

[Nagao, 1989] Nagao, M., *Machine Translation: How Far Can It Go?*, Oxford, 1989.

[Pollard and Sag, 1987] Pollard, C. and Sag, I., *Information-Based Syntax and Semantics*, CSLI Lecture Notes, 13, 1987.

[Riesbeck and Martin, 1986] Riesbeck, C. and Martin, C., "Direct Memory Access Parsing," *Experience, Memory, and Reasoning*, Lawrence Erlbaum Associates, 1986.

[Stanfill and Waltz, 1988] Stanfill, C. and Waltz, D., "The Memory-Based Reasoning Paradigm," *Proceedings of the Case-Based Reasoning Workshop*, DARPA, 1988.

[Stanfill and Waltz, 1986] Stanfill, C. and Waltz, D., "Toward Memory-Based Reasoning," *Communications of the ACM*, 1986.

[Stanfill, 1988] Stanfill, C., "Memory-Based Reasoning Applied to English Pronunciation," *Proceedings of the AAAI-88*, 1988.

[Sumita and Iida, 1991] Sumita, E., and Iida, H., "Experiments and Prospects of Example-Based Machine Translation," *Proceedings of ACL-91*, 1991.

[Tanaka and Numazaki, 1989] Tanaka, H. and Numazaki, H., "Parallel Generalized LR Parsing based on Logic Programming," *Proceedings of the First International Workshop on Parsing Technologies*, Pittsburgh, 1989.

[Tomabechi, 1987] Tomabechi, H., "Direct Memory Access Translation," *Proceeding of IJCAI-87*, 1987.

[Tomita, 1986] Tomita, M., *Efficient Parsing for Natural Language*, Kluwer Academic Publishers, 1986.

[Waltz, 1990] Waltz, D., "Massively Parallel AI," *Proceedings of AAAI-90*, 1990.