Application of the Direct Memory Access paradigm to natural language interfaces to knowledge-based systems

Hideto TOMABECHI and Masaru TOMITA
Center for Machine Translation
Carnegie Mellon University
Pittsburgh, PA 15213 U.S.A.

Abstract
This paper describes the use of the Direct Memory Access (DMA) paradigm in a practical natural language interface. Advantages and disadvantages of DMA in such applications are discussed. The DMA natural language interface "DM-COMMAND" described in this paper is being used for development of a knowledge-based machine translation system at the Center for Machine Translation (CMT) at Carnegie Mellon University.

I. Introduction
The Direct Memory Access (DMA) paradigm has been researched as a new model for natural language processing (Riesbeck & Martin [1985] and Riesbeck [1986], Tomabechi [1987a]). In this paradigm, natural language understanding is viewed as an effort to recognize input sentences by using pre-existing knowledge in memory, which is often experiential and episodic. It is contrasted with traditional models of parsing in which syntactic and semantic representations are built as the result of parsing and are normally lost after each parse. In the DMA model, input sentences are identified with the memory structure which represents the input, and are instantiated to represent that specific input. Since understanding is performed as recognition through the memory network, the result of understanding is not lost after each sentence is processed. Also, since parsing and memory-based inferences are integrated, various memory-based activities can be triggered directly through natural language understanding without separate inferential processes.

As one application of DMA, at the Center for Machine Translation (CMT) at Carnegie Mellon University, we have developed a natural language interface for our large-scale knowledge-based machine translation system1 called DM-COMMAND. This application of DMA demonstrates the power of this model, since direct access to memory during parsing allows dynamic evaluation of input commands and question answering without running separate inferential processes, while dynamically utilizing the MT system's already existing domain knowledge sources. The implementation of the DMA natural language system has been completed and is used for development of actual grammars, domain knowledge-bases, and syntax/semantic mapping rules by the researchers at CMT. This system has been demonstrated to be highly effective as a MT developmental support system, since researchers who develop these individual knowledge sources are otherwise unknowable about the internal implementation of the MT system. The DMA natural language interface can provide access (currently English and Japanese) to the system's internal functions through natural language command and query inputs. This use of the DMA model for natural language interfaces demonstrates that it is an effective alternative to other natural language interface schemes.

II. A background of DMA
The Direct Memory Access method of parsing originated in Quillian's [1968] notion of semantic memory, which was used in his TLC (Quillian [1969]) which led to further research in semantic network-based processing2. TLC used breadth-first spreading marker-passing as an intersection search of two lexically pointed nodes in a semantic memory, leaving interpretation of text as an intersection of the paths. Thus, interpretation of input text was directly performed on semantic memory. Although TLC was the first DMA system, DMA had not been explored as a model of parsing until the DMAP0 system of Riesbeck & Martin, except as a scheme for disambiguations. DMAP0 used a guided marker-passing algorithm to avoid the problem of an explosion of search paths, from which a dumb3 (not guided) marker passing mechanism inherently suffers. DMAP0 used P-markers (Prediction markers) and A-markers (Activation markers) as markers passed around in memory, adopting the notion of concept sequence to represent linear ordering of concepts as linguistic knowledge, which guides linear predictions of concepts sending P-markers in memory.

1The CMT-MT system which is the target system for the DM-COMMAND system described in this paper is described in detail in Tomita & Carbonell [1987] and Mitamura, et al [1988].

2Such as Fahlman [1979], Firth & Charniak [1982], Charniak [1983], Hockenmaier & Reiner [1983], Firth [1984], Charniak [1986], Nervig [1987], and connectionist and distributed parallel models including Smollett, et al [1982], Granger & Eiselt [1984], Waltz & Pollack [1984], Waltz & Pollack [1985], Berg [1987], and Bookman [1987].

3We call it 'dumb' when markers are passed everywhere (through all links) from a node. In a 'guided' scheme, markers are passed through specific links only.
Concept sequences, which encompasses plural patterns, are attached to nodes in memory that represent some specific experiential memory structure. In DMAP, A-markers are sent above in the abstraction hierarchy from the lexically activated node in memory, and P-markers are sent to the next element of the concept sequence only after the A-marker from below hits a node that is already P-marked. Concept refinement is performed using concept refinement links (Conf-links) when a whole concept sequence is activated. Concept refinement locates the most specific node in memory, below the activated root node, which represents the specific instance of the input text. DMTRANS (Tomabechi[1987a]) evolved the DMA into a theory of cross-linguistic translations and added mechanisms of explanatory generation, C-Marker passing (for further contextual disambiguations), and a revised scheme of concept refinement while performing English/Japanese translations.

III. DM-Command

The DM-COMMAND system which we describe in this paper is a natural language interface developed for grammar, knowledge-base, and syntax/semantic mapping rule writers at CMT, which enables these researchers to access the MT system's internal functions for their development and debugging purposes. The DM-COMMAND parser borrows the basic algorithm from the DMTRANS machine translation system, which performs recognition of input via the guided spreading activation marker-passing of A-markers, P-markers and C-markers in memory.

As a brief example, let us consider the processing of the input command "show me *HAVE-A-PAIN", where *HAVE-A-PAIN is an actual name of a concept definition in our frame system (FRAMEKIT, Nyborg[1988]). Independent of the semantic network of domain knowledge used by the MT system, the DM-COMMAND has separate memory network representing concepts involved in performing various actions in the MT system. Among such concepts is the concept 'show-frame', which represents the action of pretty-printing FRAMEKIT definitions stored as domain knowledge. This concept has the concept sequence <mirtans-word person *CONCEPT> attached to it. This concept sequence predicts that the first input word may point to an instance of 'mirtans-word' (such as 'show'), followed by an instance of person followed by some concept in the form of a FRAMEKIT name. When the first input word "show" comes in, it activates (puts an A-marker on) the lexical node 'show', which in turn sends activation (A-marker) above in the abstraction hierarchy and hits 'mirtans-word'. At the very beginning of parsing, all the first elements of concept sequences are predicted (P-marked), therefore, when an A-marker is sent from 'show' and hits 'mirtans-word', 'mirtans-word' is already P-marked. Thus, the A-marker and P-marker collide at 'mirtans-word'. When this collision of two markers happens, the P-marker is sent to the next element of concept sequence, which is 'person'. Then the next word, "me", activates the lexical node '1st person' and then activates 'person' (an A-marker is sent above in the abstraction hierarchy). Since 'person' was P-marked at a previous marker collision at 'mirtans-word', another collision occurs here. Therefore, a P-marker is again sent to the next element of the concept sequence, which is "CONCEPT". Finally, "*HAVE-A-PAIN" comes in. Now, the spreading activation occurs not in the command memory network, but in the domain knowledge network (doctor/patient dialog domain) activating "*HAVE-A-PAIN" initially and then activating the concepts above it (e.g., "*HAVE-A-PAIN" until the activation hits the concept "CONCEPT" which was P-marked at the previous collision. Since it is the final element of the concept sequence <mirtans-word person *CONCEPT>, this concept sequence is accepted when this collision of A-marker and P-marker happens. When a whole concept sequence is accepted, we activated the root node for the sequence, which in this case is the concept 'show-frame'. Also, in addition to activating this concept, we perform concept refinement, which searches for a specific node in the command network that represents our input sentence. Since it does not exist in this first parse, DM-COMMAND creates that concept.

One thing to note here is that the concept "*HAVE-A-PAIN" that is activated by input "*HAVE-A-PAIN" is not part of the memory network for the DM-COMMAND's MT system commanding concepts, instead it is a memory unit that is a part of the MT system domain knowledge, in other words "*HAVE-A-PAIN" belongs to a different memory network from 'show-frame', 'mirtans-word', and 'person'. This does not cause a problem to the DM-COMMAND, and actually, it can utilize any number of independent semantic networks simultaneously, as long as concept sequences guide passing of P-marker from one network to another. For example, the 'person' in the domain knowledge semantic network represent some generic person, whereas 'person' in DM-COMMAND command knowledge network represent persons involved in the use of the DM-COMMAND system.

* C-markers (Contextual-markers) were introduced in DMTRANS, and are propagated by mark contextually highlighted concepts in memory. DMTRANS used C-markers for word-base disambiguations through contextual marking. DMTRANS also added an explanatory generation mechanism which generates sentences in the target language for concepts that did not have a lexical entry in the target language, by explaining the concept in that target language.

3One thing to note here is that the concept "*HAVE-A-PAIN" that is activated by input "*HAVE-A-PAIN" is not part of the memory network for the DM-COMMAND's MT system commanding concepts, instead it is a memory unit that is a part of the MT system domain knowledge, in other words "*HAVE-A-PAIN" belongs to a different memory network from 'show-frame', 'mirtans-word', and 'person'. This does not cause a problem to the DM-COMMAND, and actually, it can utilize any number of independent semantic networks simultaneously, as long as concept sequences guide passing of P-marker from one network to another. For example, the 'person' in the domain knowledge semantic network represent some generic person, whereas 'person' in DM-COMMAND command knowledge network represent persons involved in the use of the DM-COMMAND system.

4Lyons[1984] has a discussion of "concept refinement" with his MGTRANS parser.

5DMTRANS, when such creation of concepts occurred the next was asked to provide the vocabulary, and thus served as a model for vocabulary acquisition as well as concept creation. In DM-COMMAND, we randomly generate names for such newly created concepts and name does not supply names for the newly created concepts.

6Actual inputs to DM-COMMAND are normally much longer and accompany multiple concept sequences; however, the basic mechanism for recognition of input is as explained here. Also, DM-COMMAND translator...
For the natural language interface to recognize what "mapping rule" means in the context of knowledge-base machine translation development as well as recognizing that "THE-DOWN-LEVEL IS A FOREGROUND definition, in order to show the syntax and semantic mapping, what that is associated with the concept in that domain (such as computer operating). Also when the next input is "And ASSIGN-ON", if the result of a parse is fast at each sentence, understanding of this sentence is impossible. (Other example is when input is "Read the output to Mr. Takodh" whose concept word-sense disambiguation must be performed to recognize 1) what "Read" means to send via that unit utility; and 2) "output" means the output of the parser, function, etc. according to the current context. These require the natural language interface to access the knowledge source of the MT system during parsing and also to recognize the input in the context of the domain knowledge; knowledge about system's internal implementations; and current discourse. DSA-MAND handles these because parsing is performed as recognition of context input with what is already known as domain knowledge and as knowledge about the system (in which it is used). Also, the result of each parse is not lost but accumulated in the active memory network.

4. Discussion

A. The DSA interface acts as mediator in order for a natural language interface to the interests of the machine translation system to work, the interface must be able to recognize the input based on what it already knows as domain specific knowledge in the area of translation and the system's own implementation. When some action is requested the interface must understand the request and respond according to what is requested, and therefore it is necessary to recognize the input within the context of the domain knowledge and current discourse, and to trigger the system's internal functions appropriately. For example, if a knowledge-base developper inputs "Show me all the mapping rules on 9TH-DOWN-LEVEL" in order to debug some conceptual bug in the knowledge-base, the natural language interface action to recognize what "mapping rule" means in the context of knowledge-base machine translation development as well as recognizing that "THE-DOWN-LEVEL IS A FOREGROUND definition, in order to show the syntax and semantic mapping, what that is associated with the concept in that domain (such as computer operating). Also when the next input is "And ASSIGN-ON", if the result of a parse is fast at each sentence, understanding of this sentence is impossible. (Other example is when input is "Read the output to Mr. Takodh" whose concept word-sense disambiguation must be performed to recognize 1) what "Read" means to send via that unit utility; and 2) "output" means the output of the parser, function, etc. according to the current context. These require the natural language interface to access the knowledge source of the MT system during parsing and also to recognize the input in the context of the domain knowledge; knowledge about system's internal implementations; and current discourse. DSA-MAND handles these because parsing is performed as recognition of context input with what is already known as domain knowledge and as knowledge about the system (in which it is used). Also, the result of each parse is not lost but accumulated in the active memory network.

3. Integration of Inference

The power of a natural language interface to an MT system needs to recognize the input according to what the MT system already knows as the knowledge source and according to its own internal implementations. A traditional integrated parser will require an external inferential process that will perform the tasks of context disambiguation and inference in searching for the appropriate action determined by the system's particular inferential architecture. Ideally, the inference module and the parser need interact during parsing, due to the constraints put on the understanding of the system within the context established by the knowledge domain and the system's implementation. However, unless memory and inference are integrated, such an interaction is difficult to perform, and without such interactions, parsing can either very slow or fail in contextually difficult sentences because of the interdependencies of concept meanings expressed in each language.

In the DSA-MAND system, memory is organized so that
that the concept which represents the request for action is directly connected to the concept that represents the action that is requested. Likewise, the direct memory access recognition of a question means that the concept which is identified by the input is directly connected to the concept that represents the answer, as long as the system knows (or potentially knows) the answer. In other words, in the DMA model, recognition of a request for action is a triggering of the action requested and recognition of a question is knowing the answer (i.e., as soon as we understand the question, either we know the answer, or we know the inferences to be performed (or functions to be evaluated) to get the answer) as long as memory contains the action and the answer. To reiterate the literature on the DMA paradigm, in this model, memory is organized in the hierarchical network of concepts which are related by links that define the concepts. Thus, as soon as we identify the input with a certain concept in the memory, we can trigger the action (if this is a concept that represents some action (or request for action)), or answer the question (if the concept represents some knowledge (or request for some knowledge)). Thus, parsing and inference are integrated in the memory search process, and no separate inferential modules are necessary. It should be understood; however, that it is not our claim that we can eliminate inference altogether. Our claim is that 1) the memory search through concept refinement itself is an inference which is normally performed by separate inference modules (such as contextual inference and discourse analyses modules) in other parsing paradigm; and 2) whenever further inference is necessary, such inference can be directly triggered after concept refinement from the result of parse (for example, as a daemon stored in the abstraction of the refined concept) and therefore, the inference is integrated in the memory activity.

C. Ellipsis and anaphora

In a practical natural language interface, the capacity to handle elliptic and anaphoric expressions is important. DM-COMMAND is capable of handling these phenomena, because under the DMA paradigm (which is typically called “recognize-and-record paradigm”), the result of each parse is not lost after each sentence, but instead remains as part of the contextual knowledge in the memory network. On the other hand, in the traditional parsing paradigm (we call it “build-and-store” paradigm), since the result of the parse is lost after each sentence, the parsers can at best handle indexicality within a sentence. Specifically, 1) ellipses are handled by DM-COMMAND; since ellipses are characterized as the lack of elements in a concept sequence, and these are recoverable as long as the elements or their descendants had been activated in previous parses14; 2) anaphoric and pronoun references are resolved by utilization of both semantic knowledge (represented as restrictions on possible types of resolutes) and also by the context left from the previous parses in memory similar to the way that the elliptic expressions are handled. Finding a contextually salient NP corresponding to some NP means, in DMA, searching for a concept in memory which is previously activated and can be contextually substituted for currently active concept sequence15.

D. DMA and syntax

One weakness of current implementations of the DMA paradigm is that the concept sequence is the sole syntactic knowledge for parsing16. Therefore, a DMA system needs deliberate preparation of concept sequences to handle syntactically complex sentences (such as deeply embedded clauses, small causes, many types of sentential adjuncts, etc.). This does not mean that it is incapable of handling syntactically complex sentences, instead it means that concept sequences at some level of abstraction (at syntactic template level down to phrasal lexicon (Becker[1975]) level) must be prepared for each type of complex sentence. In other words, although such sentences can be handled by the combination of concept sequences, designing such sequences can be complex and less general than using external syntactic knowledge17. Thus, current reliance upon a linear sequence of concepts causes limitations on the types of sentences that can be realistically handled in DM-COMMAND. Of course, there is nothing to prevent DMA paradigm to integrate syntactic knowledge other than a linear sequence of concepts. Actually, we have already implemented two alternative schemes for integrating phrase-structure rules into DMA. One method we used was having syntactic nodes as part of the memory and writing phrase-structure rules as concept sequences18. Another method was to integrate the DMA memory activity into an augmented context-free grammar unification in a generalized LR parsing. Second method used in a continuous speech understanding is described in Tomabechi&Tomita[ms]. We will not discuss these schemes in this paper.

While handling syntactically complex sentences is rather expensive for DM-COMMAND, since it relies solely on linear concept sequences, natural language interfaces are one appli-

---

14 For example, with the input “jgr92.gra o uchidae. sem.tst mo.” (Print jgr92.gra. Sem.tst also). Second sentence has the object dropped; however, this can be supplied since the memory activity after the first sentence is not lost and the memory can supply the missing object.
15 For example in “Pretty-print dm.lisp. Send it to mt@.nrr”, “it” can be identified with the concept in memory that represents dm.lisp which was activated in memory during the understanding of the first sentence.
16 Although generation is normally helped by external syntactic knowledge such as in the case of DMTRANS.
17 Also, pronoun and anaphora resolution is based upon contextual knowledge alone; however, use of syntactic knowledge (such as the governing category of an anaphora) would help such efforts.
18 Due to recursive nature of phrase-structure rules, we did not find this method appealing, unless we obtain a truly parallel machine.
cation area where the capacity to handle phenomena such as ellipsis, anaphora, pronoun resolution, and contextual disambiguation is more valuable than handling syntactically complex sentences. It seems that DMA is one ideal paradigm in this area. This is evident if we consider the fact that input to a natural language interface is normally in a form of dialog and users tend to input short, elliptic, ambiguous and even ungrammatical sentences to the interface. Our experience shows that an increase in the size and complexity of the system in order to integrate full syntactic processing, enhancing the DMA's capacity to handle syntactically complex sentences, has so far outweighed the need for such capacity.

V. Conclusion

DM-COMMAND is the first practical application of the DMA paradigm of natural language understanding, in which parsing and memory-based inference is integrated. This system has been proven to be highly effective in knowledge-based MT development. It is due to the complexity of system implementations in a large scale MT project that grammar and knowledge base writers are not expected to have expertise on the internals of the translation system, whereas it is necessary for such a group of project members to access the system internal functions. DM-COMMAND makes this access possible through a natural language command and question answering interface. Since DM-COMMAND uses the spreading activation guided marker-passing algorithm, in a memory access parser which directly accesses the MT system's already existing network of concepts, inference is integrated into memory activity. Since there is a separate memory network for concepts representing commanding and question-answering that are generic to MT system development, the system is highly portable. The DM-COMMAND system demonstrates the power of a direct memory access paradigm as a model for a natural language interface, since understanding in this model is a recognition of the input sentence with the existing knowledge in memory, and as soon as such understanding is done, the desired command can be directly triggered (or the question directly answered).

With DMA's ability to handle extra-sentential phenomena (including ellipsis, anaphora, pronoun reference, and word-sense ambiguity), which are typical in a practical natural language command/query inputs, DMA is one ideal paradigm for natural language interfaces as shown in our DM-COMMAND system. Also, DMA's integration of parsing and inference into an unified semantic memory search has proven to be highly effective in this application.

Appendix: Implementation

The DM-COMMAND system has been implemented on the IBM-RT and HP9000 AI workstations, both running
CommonLisp. The system directly utilizes the FrameKit-represented domain knowledge (currently in the area of computer manuals and doctor/patient conversations) of the CMU-MT knowledge-based large-scale machine translation system. It handles inputs in both English and Japanese. The current size of the DM-COMMAND system is roughly 5,000 lines of Lisp code (this does not include the MT system functions and the FrameKit frame system, parts of which must also be loaded into memory) and is not expected to increase, since the future variety in types of commands and questions that the system will handle will be integrated into the network of memory that represents concepts for commanding and question-answering and not into the system code itself²⁴. Compiled code on IBM-308 and HP9000/60 is fast enough that parsing and performing commanded action happens virtually in real-time. We are expecting to increase the variety in types of system functions and grammar/rule development functions; however, as noted above, since such increases will occur in the memory network, as a system implementation, DM-COMMAND is a completed system.

Acknowledgments

The authors would like to thank members of the Center for Machine Translation for fruitful discussions. Eric Nyberg and Teruko Mitamura were especially helpful in preparing the final version of this paper.

References

[1] Becker, J.D. (1975) The phrasal lexicon. In 'Theoretical issues in natural language processing'. Proceedings of the workshop of the ACL. Eds. Schank, R.C. and Nash-Webber, B.N.
[2] Berg, G (1987) A Parallel Natural Language Processing Architecture with Distributed Control. In 'Proceedings of the CogSci-87'.
[3] Bookman, L.A. (1987) A Microfeature Based Scheme for Modeling Semantics. In 'Proceedings of the IJCAI-87'.
[4] Charniak, E. (1983) Passing Markers: A theory of Contextual Influence in Language Comprehension. Cognitive Science 7.
[5] Charniak, E and Santos, E (1987) A Connectionist Context-Free Parser Which is Not Context-Free, But Then It Is Not Really Connectionist Either. In 'Proceedings of the CogSci-87'.
[6] Charniak, E (1986) A neat theory of marker passing. In 'Proceedings of the AAAI-86'.
[7] Collingford, R.E. and Booth, S.L. (1985) How to make a natural-language interface robust. MIT-ICS-85/21, Georgia Institute of Technology.
[8] Fahlman, S.E. (1983) NETL: A system for representing and using real-world knowledge. The MIT Press.
[9] Granger, R.H., Elshel, K.P. (1984) The parallel organization of lexical, syntactic, and pragmatic inference processes. In 'Proceedings of the First Annual Workshop on Theoretical Issues in Conceptual Information Processing'.
[10] Hahn, U. and Reimer U. (1983) World expert parsing: An approach to text parsing with a distributed lexical grammar. Technical Report, Universität Konstanz, West Germany.
[11] Hirst, G. and Charniak, F. (1982) Word Sense and Slot Disambiguation. In 'Proceedings of AAAI-82'.
[12] Hirst, G. (1984) A Semantic Process for Syntactic Disambiguation. In 'Proceedings of AAAI-84'.
[13] Lytinen S. (1984) The organisation of knowledge in a multi-lingual, integrated parser. Ph.D. thesis Yale University.
[14] Mitamura, T., Musha, H., Kee, M. (1988) The Generalized LR Parser/Compiler Version 8.1: User's Guide, ed. Tomita, M. CMU-CMT-88-MEMO. Carnegie Mellon University.
[15] Norvig, P. (1987) Inference in Text Understanding. In 'Proceedings of the AAAI-87'.
[16] Nyberg, E. H. (1988) The FRAMEKit User's Guide Version 2.0. CMU-CMT-MEMO, Carnegie Mellon University.
[17] Quillian, M.R. (1968) Semantic Memory. In 'Semantic Information Processing', ed. Minsky, M. MIT Press.
[18] Quillian, M.R. (1969) The teachable language comprehender. BBN Scientific Report 10.
[19] Riesbeck, C. (1975) Conceptual Analysis. In 'Conceptual Information Processing' ed. Schank, R. C. North Holland.
[20] Riesbeck, C. (1986) From Conceptual Analyzer to Direct Memory Access Parsing: An Overview. In 'Advances in Cognitive Science 1' ed. Sharkey, N.E. Ellis Horwood.
[21] Riesbeck, C. and Martin, C. (1985) Direct Memory Access Parsing. Yale University Report 354.
[22] Small, S., Cottereill, G. and Shastri, I. (1982) Toward connectionist parsing. In 'Proceedings of the AAAI-82'.
[23] Tomabechi, H. (1987a) Direct Memory Access Translation. In 'Proceedings of the IJCAI-87'.
[24] Tomabechi, H. (1987b) Direct Memory Access Translation: A Theory of Translation. CMU-CMT-87-105, Carnegie Mellon University.
[25] Tomabechi, H. and Tomita, M. Manuscript. The Integration of Unification-based Syntax/Semantics and Memory-based Pragmatics for Real-Time Understanding of Noisy Continuous Speech Input.
[26] Tomita, M. and Carbonell, J. (1987) The Universal Parser Architecture for Knowledge-Based Machine Translation. In 'Proceedings of the IJCAI-87'.
[27] Waltz, D.L. and Pollack, J.B. (1984) Phenomenologically plausible parsing. In 'Proceedings of the AAAI-84'.
[28] Waltz, D.L. and Pollack, J.B. (1983) Massively Parallel Parsing: A Strongly Interactive Model of Natural Language Interpretation. Cognitive Science 9.