TEXT UNDERSTANDING WITH MULTIPLE KNOWLEDGE SOURCES: AN EXPERIMENT IN DISTRIBUTED PARSING

Cinzia Costantini, Danilo Fum, Giovanni Guida,
Angelo Montanari, Carlo Tasso

Laboratorio di Intelligenza Artificiale
Dipartimento di Matematica e Informatica
Universita' di Udine
Udine, Italy

ABSTRACT

A novel approach to the problem of text understanding is presented, which exploits a distributed processing concept, where knowledge from different sources comes into play in the course of comprehension. In the paper the rationale of advocating such an approach and the advantages in following it are discussed. A prototype parser based on an original distributed problem-solving architecture is presented. It encompasses a centralized declarative control module and a collection of decentralized, loosely coupled, heterogeneous problem solvers specialized in the various facets of the parsing task. The mechanisms of coordination and communication among the specialists are illustrated, and an example of the parser operation is given. The parser is implemented in LISP on a SUN workstation.

1. INTRODUCTION

The processes underlying text understanding involve a variety of complex, multifaceted activities which have not been yet completely understood from the cognitive point of view, and which still lack adequate computational models. Recent research trends in cognitive science and artificial intelligence, however, have put forward some ideas concerning human cognition and automatic problem solving that offer promising tools for the design of text understanding systems.

One of the key ideas emerged in the field of cognitive study of natural language comprehension is that text understanding constitutes in humans an interactive process, where bottom-up, data-driven activities combine with top-down, expectation-driven ones to cooperatively determine the most likely interpretation of the input (Lesgold and Perfetti, 1981). Roughly speaking, humans begin with a set of expectations about what information is likely to be found in the text. These expectations are based both on linguistic knowledge (about words, phrases, sentences, and larger pieces of discourse) and on non-linguistic world knowledge. As information from the text becomes available, the reader strengthens those hypotheses that are consistent with the input and weakens those that are inconsistent. The stronger hypotheses, in turn, make even more specific predictions about the information represented in the text, so as the initial expectations are successively corrected and refined until they eventually yield an adequate approximation of the meaning of the text.

In one of the first and more detailed descriptions of interactive processes in text understanding, Rumelhart (1977) has proposed a model comprising several knowledge sources, each one operating independently and in parallel with the others. These knowledge sources are processors operating at different levels of linguistic representation. The outputs of each of these knowledge sources are hypotheses or best guesses from the data available at that level. The hypotheses are transferred to a central device, called the message center, where they can be observed by all other knowledge sources, thus being available as evidence for or against hypotheses at other levels. In a more dynamic view of interaction, Levy (1981) suggests that the message center could modify the activity of each individual processor. That is, when a particular hypothesis has strong outside support, the analyzers of a particular knowledge source may change their own processing either to seek confirming evidence for it or to accept that view and therefore stop analyzing information that would otherwise have been tested.

The idea of decomposing a difficult problem into a large number of functionally distinct subproblems, each one being tackled by a specialized problem solver, has been pursued with great interest in the last years also in the field of artificial intelligence, where the area of distributed problem solving has developed into a much researched and hot topic. Several computational paradigms have been proposed, such as blackboard systems (for a review, see: Nii, 1986a; 1986b), contract net (Davis and Smith, 1983), the scientific community metaphor (Kornfeld and Hewitt, 1981), FA/C systems (Lesser and Corkill, 1981) which proved appropriate to several tasks and application domains. As far as the field of text understanding is concerned, we mention here the work of Cullingford (1981) on DSAM, the distributed script applier, in which an arbitrary number of distinct, potentially distributed, processors are used to read and summarize newspaper stories.

In this paper we present a novel approach to the problem of text understanding through a distributed processing paradigm, where different knowledge sources come into play and cooperate in the course of comprehension. In section two we deal with the rationale of advocating such an approach and the advantages (and disadvantages) in following it. Section three illustrates the general architecture of a prototype distributed parser, and describes the mechanisms of coordination and communication among the various knowledge sources. In section four we present an example of the parser operation through the tracing of the analysis of a sample sentence. Finally, section five deals with the current state of the implementation and highlights the novelty and originality of the approach.

2. RATIONALE AND TECHNICAL REQUIREMENTS

Several reasons recommend and support the choice of a distributed approach to text understanding. From a cognitive point of view, it is indubitable that humans perform such an activity incrementally. That is, not all what can be derived from a text becomes evident since the very beginning. Some features of the text are understood almost automatically and with minimum effort, others require more labor, whereas still others become clear only after a thoughtful process. This increasing depth of processing, which has differential effects about what is understood from the same piece of text

should be modeled also in an automatic system. A distributed architecture comprising a collection of specialized problem solvers (specialists) with different skill and competence, and working with different knowledge sources, seems a promising way to achieve incrementality.

Such an architecture offers several advantages from a technical point of view, too. About these we mention:

- the possibility to adopt different techniques and methodologies for each specialist,
- the fact that specialists can be developed in isolation independently from each other;
- the possibility to change one or more specialists without implying a global restructuring of the whole system,
- the robustness that can be achieved by overlapping the capabilities of different specialists.
- the facility in designing and debugging,

The main problem in adopting a distributed approach is that of control, i.e. making the specialists cooperate. As Cullingford (1981: 52) puts it, "... In an ideal system each expert would become available only when needed, run only so long as it had something useful to do and communicate its findings to interested parties in an efficient manner. If an appropriate level of integration could be achieved, one could hope to improve the capabilities of an understanding system by adding new knowledge sources, to reuse experts in different problem domains and to investigate the relative performance degradation due to removing various knowledge sources.".

In our approach we adopt a form of control based on the interaction of each individual specialist with a central manager, which supervises and directs the overall operation of the system by coordinating the autonomous activities of the specialists (bottom-up approach), and by exploiting its own general problem solving strategies (top-down approach).

The prototype distributed parser which has been developed according to the ideas outlined above works in the domain of descriptive text understanding, more precisely computer science literature on operating systems. It receives in input a natural language text and produces in output a semantic representation of its meaning in the BLR/ELR representation language (Fum, Guida, and Tasso, 1984). Three main objectivers have been taken into account in the design of the parser:

- *Incrementality of parsing and generation of the BLR/ELR.* As the parser has to cover a large variety of linguistic features and must rely upon a number of different knowledge sources, it seems appropriate that both analysis of the input text and generation of the BLR/ELR representation are carried out in a stepwise manner through successive additions and refinements. Also the structure itself of the BLR/ELR formalism, made up of a collection of propositions appropriately connected together and supplemented with additional information (e.g., about time, quantification, etc.), strongly suggests an incremental approach to parsing.
- *Cognitive validity.* The parser should not only produce a correct BLR/ELR representation of the input text, but it should also show some degree of linguistic competence in the way it operates internally. In other words, it should provide an acceptable approximation of the basic mental processes that occur in humans.
- *Effectiveness.* The parser should be capable of operating in an efficient and correct way in non-trivial cases. Moreover, the parser should be easy to design and debug.

3. A DISTRIBUTED ARCHITECTURE

3.1 Overall System Architecture

As mentioned above, our distributed parser is constituted by a collection of individual *specialists*, each one expert in a facet of the parsing problem (e.g., syntactic analysis, disambiguation, reference, semantics, time, quantification, BLR/ELR construction, etc.). Each specialist is an autonomous problem solver, which has its own competence domain, where it can operate with certain and complete knowledge. However, it is assumed that no specialist has enough knowledge and competence to cover the whole parsing activity: all (or most) of them are necessary to successfully complete the parsing of a complex text. Moreover, we assume that specialists may be heterogeneous, i.e., implemented using different technologies (e.g., a deterministic algorithm, a knowledge-based system, etc.). Also, they may have partially overlapping competence areas, and even be redundant, i.e. there may be several specialists for the same task (e.g., for syntactic analysis). As we have stated that specialists are independent problem solvers, we also assume that they have no mutual knowledge: they do not know about each other, they do not even know about the existence of other specialists. This assumption is very important to allow a fully independent design of an individual specialist, without bothering about the others.

Each specialist can solve a well defined class of problems, and once a problem has been assigned to it, it can result in three different outcomes:

- *success*, i.e. the problem assigned has been solved and its solution produced;
- *fail*, i.e. the specialist has been unable to solve the problem and an alarm message is returned;
- *need-help*, i.e. the specialist has been successful in decomposing and partially solving the problem at hand, but it needs help from outside to proceed further in the solution process. In this case, the current problem is suspended, and (sub)problems are generated for which solutions are needed.

The internal operation of each specialist is not of interest here, as we have assumed that they may be heterogeneous. What is crucial is the interface they show towards the outside which is expected to be very simple. A specialist may receive a problem to solve, and issue a solution, other problems, or an alarm. It may also receive a solution to one of the (sub)problems it has previously generated, which will be used to resume the solution process of some suspended problem.

3.2 Communication and Control Mechanisms

Specialists are not allowed to directly communicate to each other, but can only communicate to a *cooperation manager*, which is in charge of organizing and controlling the overall activity of the parser. It embodies knowledge about:

- the actual architecture of the system, i.e. how many and which specialists are available;
- the competence of each individual specialist;
- how to match problems to specialists in order to exploit in the best way their specific capabilities;
- how to schedule the activity of the specialists, i.e. which specialists to activate first, taking into account priority and redundancy problems;
- how to correctly switch messages among specialists.

The communication between specialists and the cooperation manager occurs according to a fixed protocol which includes three basic types of messages, namely: *problems*, *solutions*, and *alarms*, as already outlined above.

The working memory of the parser is a partitioned shared memory, where each specialist can read and write in its own partition only, but has full visibility on the entire memory. Clearly, in order to allow specialists to work correctly on the shared memory, it is necessary that a common representation language is adopted, at least for information that may concern more than one specialist.

The operation of the cooperation manager is basically message-driven: it is all the time waiting for messages and, as soon as messages arrive from the specialists, they are stored in a buffer and later examined and treated according to some specific policy (e.g., the priority of the messages or their origin may be taken into account). The cooperation manager is in charge of three main activities:

- it assigns problems to specialists according to their competence, current work load, etc.;
- it passes solutions to the relevant specialists (i.e., those who issued the (sub)problem to which the solution refers);
- it manages alarms (e.g., by resorting to alternative specialists with similar or overlapping competence).

The cooperation manager, however, in addition to the above mentioned message handling capability, has also its own strategies that can override, when needed, the basic message-driven style, thus affecting the overall operation of the parser. These strategies, that embed knowledge about "how to manage the parsing task", are crucial to the successful activity of the parser if we really want to allow individual specialists to be designed and constructed independently from each other. In fact, as no global strategy is coded in the system, it must be explicitly assigned as an additional competence to the cooperation manager.

3.3 The Specialists

As illustrated above, our distributed parser is well suited to host a large variety of specialists. We will briefly list in the following some of those utilized in the current implementation of the system.

- The *Morphology Specialist (MS)* is devoted to perform the morphological analysis of each word, i.e. extracting from the Dictionary all the relevant information and determining the appropriate morphological types and variables.
- The *Encyclopedia Specialist (ES)* is able to access the Encyclopedia for extracting semantic information and world knowledge.
- The *Syntax Specialist (SYS)* is able to identify the constituents of a sentence and to build up a parse tree. The current version is implemented through a context-free grammar augmented with transformational rules.
- The *Semantics Specialist (SES)* is devoted to a semantic analysis of a sentence performed only through semantic information, discarding any syntactic processing.
- The *Syntax-Semantics Specialist (SSS)* is able to complement semantic analysis with available syntactic information (and vice-versa) in order, for example, to resolve ambiguities.
- The *Time Specialist (TS)* is able to attach to each proposition of the BLR/ELR the appropriate temporal information.
- The *Reference Specialist (RS)* is devoted to analyze pronominal and anaphoric references.
- The *Quantification Specialist (QS)* is capable of identifying the appropriate quantifier to attach to each concept in the BLR/ELR.
- The *BLR/ELR Generator Specialist (BEGS)* is devoted to integrate all the information useful to actually build up the BLR/ELR representation of the meaning of the text.

4. EXPERIMENTAL RESULTS

In this section we will shortly illustrate some of the most significant characteristics of the parsing process by means of the analysis of a simple sentence extracted from a text on operating systems. Let us consider the following fragment of text:

"... An integer priority is assigned by the scheduler to each process in the ready-queue. ..."

The Cooperation Manager, hereinafter CM, is devoted to organize the work of the specialists which are able to solve specific parts of the overall problem. In the current version, the CM largely relies on the BEGS specialist for structuring the parsing process and for generating the BLR/ELR: each sentence in the text is processed one after the other, from left to right. We will discard in this illustration all the details concerning this specialist, as well as other specialists which are not essential for understanding the system operations. Moreover, we will not describe how the management of the shared memory and its partitions is actually carried on.

As already mentioned, the CM can implement several parsing strategies by forcing different ways of organizing the contribution of the specific specialists to the solution of the overall task. It is important to stress that the proposed architecture allows to change quite easily the strategy adopted. In this example, a semantics-directed parsing will be shown. More specifically, when the sample sentence shown above is considered, the CM will assign the problem of semantically analyzing the sentence to all the specialists potentially capable to perform such an analysis (in the current version, SES and SSS). At the same time, it will assign to QS, RS, and TS the quantification, reference, and temporal analysis task, respectively. Appropriate problem messages will be sent to each of them, such for example:

To: SES
From: CM
Problem: Semantic Analysis
On: < ... the current sentence ... >
Priority: Auxiliary.

Also the message requesting semantic analysis from SSS will have an Auxiliary priority since for the same problem more than one specialist is engaged and can possibly find a correct answer. When, later on, one of them will possibly recognize its inability to correctly complete the task, it will send back to the CM a message containing an alarm, causing in such a way a change in the priority of the semantic analysis problem, that will become Fatal. The other three messages sent by the CM to RS, QS, and TS will have a Fatal priority, because no alternative specialists are able to contribute to the solution of that part of the overall problem.

After these initial problem assignments, the CM enters a suspended state, which will be resumed whenever messages from any of the specialists will be received. RS, QS, and TS can generally carry on their activity only after some semantic information about the sentence has been provided. To this purpose, all these three specialists will send to the CM a message of the kind

```
To:      CM
From:    ...
Problem: Semantic Analysis
On:      < ... the current sentence ... >
Priority: Fatal.
```

The use of a Fatal priority will cause a synchronization of the three specialists with the completion of semantic analysis, since their activity will be suspended as long as they will not receive back from the CM a solution message containing an answer to this problem. In this case, CM has already sent appropriate requests concerning the semantic analysis, and therefore all the activities will remain suspended until completion of the task.

As noted above, both the SES and SSS specialists are called to give their contribution to semantic analysis. The first that will come up with a complete solution will allow the CM to answer RS, QS, and TS.
In this specific case SES is able to answer only partially. Semantic information on the concepts in the sentence will be requested through a problem message that the CM will forward to ES. The information that ES is able to extract from the Encyclopedia will include the following fragments:

```
(INTEGER
    (Relation INTEGER (VALUED-THING))
    ...
(PRIORITY
    (Entity PRIORITY
    (Is-a COMPARABLE-THING, VALUED-THING,
        ASSIGNABLE-THING, ...)
    ...
(ASSIGN
    (Relation ASSIGN (ASSIGNER, ASSIGNABLE-THING,
                      ASSIGNEE))
    (Arg-Roles
        (ASSIGNER Subject)
        (ASSIGNABLE-THING Object)
        (ASSIGNEE To))
    ...
(SCHEDULER
    (Entity SCHEDULER)
    (Is-a PROCESS ... ))
    ...

(PROCESS
    (Entity PROCESS)
    (Is-a PROGRAM, MODIFIER, ASSIGNER, ASSIGNEE, ... ))
    ...
```

Through Is-a inheritance, SES will correctly infer that INTEGER is predicated about PRIORITY, but also that both the concepts SCHEDULER and PROCESS could correctly instantiate both the first and the third argument of ASSIGN. This allows construction of the following BLR piece:

```
10 ASSIGN ( ? , /PRIORITY/, ? )
20 INTEGER (/PRIORITY/)
```

where the slash indicates that neither quantification, nor reference or temporal information are included yet in the BLR.
As syntactic information is not taken into account, SES sends an alarm message to the CM, since unable to build up the complete solution.
On the other hand, SSS will succeed by integrating syntactic information provided by SS, and the semantic information shown above. SS needs also morphological information contained in the Dictionary, that will be requested through an appropriate message to the CM. The outcome produced by SSS is a more complete version of the BLR, containing:

```
10 ASSIGN (/SCHEDULER/, /PRIORITY/, /PROCESS/)
20 INTEGER (/PRIORITY/)
30 LOC (/PROCESS/, /READY-QUEUE/),
```

where the ambiguity of considering READY-QUEUE as an argument of ASSIGN or as an argument of the predicate LOC (relative to the preposition "in") has been resolved by means of semantic agreement between predicates and arguments. This solution will allow the CM to send an answer to the three suspended specialists QS, RS, and TS, that will resume their operation.

It is interesting to illustrate how QS, RS, and TS can cooperate together.
QS starts its processing from the logical subject of the sentence, i.e. SCHEDULER. In order to determine whether the definite article should be considered as indicating an anaphoric reference or something else, it will send the following problem message to the CM:

```
To:      CM
From:    QS
Problem: Referent-of
On:      SCHEDULER
Priority: Fatal
```

that will be forwarded to RS. Two things could happen at this point: the concept was already mentioned in previous parts of the text and RS will send back the corresponding identifier as a solution, or the concept was not mentioned before in the text, and RS will answer that this is the first occurrence of SCHEDULER. In the former case, QS will quantify the entity with an existential quantification. In the latter case, the need arises of considering also the tense of the verb, that can be provided by TS. The present tense of "is assigned" makes QS decide for a universal quantification (Hess, 1985), i.e. "every scheduler assigns a priority".
Assuming the latter interpretation, QS will continue its processing with READY-QUEUE. Again RS will check whether a previous reference exists. Since this is not the case, RS looks for implicit references. The ES can provide an answer to this request, since in the SCHEDULER frame of the Encyclopedia it is stated that "schedulers are associated with waiting-queues, ready-queues, etc.". READY-QUEUE is then considered to be one of the ready-queues associated with SCHEDULER. Moreover, since scheduler is already universally quantified, it will result that READY-QUEUE is existentially quantified with respect to SCHEDULER, i.e. "for every scheduler there exists a ready-queue".
This kind of process is carried on until eventually BEGS will integrate all the contributions of the other specialists, producing the following BLR/ELR:

```
10 ASSIGN (SCHEDULER:X0, F0(X1), TYPE0:X1, PERM)
20 *PRIORITY (F0(X1))
30 INTEGER (F0(X1))
40 $DEFINE (TYPE0, LAMBDA (Z0))
50 *PROCESS (Z0)
60 LOC (Z0, F1(X0))
70 *READY-QUEUE (F1(X0)).
```

An important aspect of the operation of CM is worthy to be stressed again: all the problem messages that CM receives do not contain any explicit suggestion on the specialist(s) that should be invoked. It is specific responsibility of the CM to manage these assignments by means of a specific knowledge base devoted to this task. In the current version of the system this is implemented through a simple rule-based mechanism.

5. CONCLUSIONS

In the paper we have presented a novel approach to text understanding which is supported by an experimental parser based on a distributed architecture. The originality of this approach consists in utilizing a shared memory and a centralized control for managing a distributed processing environment. This allows implementation of a very flexible behavior, resulting from the dynamic interaction between a supervisor (the Communication Manager), which elaborates heuristic strategies involving several knowledge sources, and a set of independent specialists which individually contribute to the overall problem solving process. The parser works in the domain of computer science literature on operating system and has been implemented in a prototype version in Franz-LISP on a SUN 2 workstation.

References

Cullingford, R.E. (1981) Integrating Knowledge Sources for Computer "Understanding" Tasks. *IEEE Transactions on Systems, Man, and Cybernetics,* 11: 52-60.

Davis, R. and Smith, R.G. (1983) Negotiation as a Metaphor for Distributed Problem Solving. *Artificial Intelligence,* 20: 63-109.

Fum, D., Guida, G., and Tasso, C. (1984) A Propositional Language for Text Representation. In: B.G. Bara and G. Guida (Eds), *Computational Models of Natural Language Processing,* Amsterdam,NL: Elsevier.

Hess, M. (1985) How Does Natural Language Quantify? *Proc. Second Conference. of the European Chapter of ACL,* Geneva, CH: 8-15.

Kornfeld, W.A. and Hewitt, C.E. (1981) The Scientific Community Metaphor. *IEEE Transactions on Systems, Man, and Cybernetics,* 11: 24-33.

Lesgold, A.M. and Perfetti, C.A. (1981) *Interactive Processes in Reading.* Hillsdale, NJ: Erlbaum.

Lesser, V.C. and Corkill, D.D. (1981) Functionally Accurate, Cooperative Distributed Systems. *IEEE Transactions on Systems, Man, and Cybernetics,* 11: 81-96.

Levy, B.A. (1981) Interactive Processes during Reading. In: A.M. Lesgold and C.A. Perfetti (Eds), *Interactive Processes in Reading.* Hillsdale, NJ: Erlbaum.

Nii, H.P. (1986a) Blackboard Systems: The Blackboard Model of Problem Solving and the Evolution of Blackboard Architectures. *AI Magazine,* 7: 38-53.

Nii, H.P. (1986b) Blackboard Systems: Blackboard Application Systems, Blackboard Systems from a Knowledge Engineering Perspective. *AI Magazine,* 7: 82-106.

Rumelhart, D.E. (1977) Toward an Interactive Model of Reading. In: S. Dornic and P. Rabbitt (Eds.), *Attention and Performance VI,* Hillsdale, NJ: Erlbaum.