A Cooperative Model of Intuition and Reasoning for Natural Language Processing – Microfeatures and Logic

Hideo Shimazu & Yosuke Takashima
C&C Information Technology Research Laboratories
NEC Corporation

ABSTRACT

This paper discusses problems of right retrievals of memory, preferential orderings, and script selection/withdrawal in natural language processing (NLP). **Atmosphere** is introduced to solve these problems. It works as a contextual indicator which roughly grasps what is being talked about. An implementation mechanism for **atmosphere** is presented inspired by artificial neural network researches. It is characterized by microfeature representation, a chronological FIFO (First-In First-Out memory), and threshold-based selection. The mechanism constructs an intuition module and works for NLP while cooperating with a logic module which uses TMS to check the justifications of preferential decisions done by the intuition module.

MOTIVATION

A problem solving task can be divided into two paradigms; *solving by reasoning* and *solving by intuition*. Natural language processing (NLP) is also divided into these two solving paradigms. Examples of *solving by intuition* in NLP are followings:

(Ex-1) **Right memory retrieval:**

While we talk about tennis if we call a name like Ron, we can extract a memory structure corresponding to the right *Ron* among many Rons we know.

(Ex-2) **Preferential ordering:**

S1. The astronomer married a star. She lived in Hollywood.

S1 is a modification of an example in [1]. When we read S1, we naturally interpret that a male astronomer married a movie star who lived in Hollywood. But there is another logically correct interpretation; a female astronomer living in Hollywood married a male movie star. Such a preferential ordering is what humans do and AI programs don't do.

Right memory retrievals and preferential orderings are important tasks in NLP. However, AI has not established a proper mechanism which deals with them. We introduce **atmosphere** to deal with them. **Atmosphere** is a contextual indicator which roughly grasps what is being talked about now. By introducing **atmosphere**, the followings are achieved:

- **Atmosphere-based memory retrieval:**
  Right memory retrievals and preferential orderings are achieved.

- **Atmosphere-based script selection/withdrawal:**
  Script [16] is an important knowledge structure to express contextual information. A difficult problem is to select/withdraw scripts at the right time and place. If each script is pre-defined its typical atmosphere, distances between the current contextual atmosphere and atmosphere definitions in each script can be calculated. If the atmosphere of a script is closely approximate to that of the current context, such a script can be thought of a proper script at the place and time. The comparison of atmospheres can also be used to decide when to withdraw scripts which are now being selected.

In the following section, the strategy and approach of our research are described. Next, a microfeature based realization of **atmosphere** inspired by artificial neural network (ANN) researches is presented. Then, a cooperative model of an intuition module and a logic module is described. Finally, the implementation details of the model is presented.

STRATEGY AND APPROACH

Recently artificial neural networks (ANN) [15] are paid attention as the first rival against AI. Also in natural language processing fields, many researchers are doing researches using ANN [13][17] [9] [8]. ANN seems a good candidate to handle **atmosphere** since it has many good features AI does not have. For example, Rumelhart et al. proposed a completely new point of view towards schema definition [14]. However, ANN still has many hard problems, some of which can be dealt with by AI. They are variable bindings, schema/role bindings, recursive structures, instantiations, inheritance, one-shot learning, sequential input, etc [7].

In addition to these problems, from the practical implementation point of view, ANN is hard to develop practical programs at the current stage. A schema concept is abstract and does not have an explicit boundary as a module [14]. It is hard to define, modularize, maintain and change such knowledge structures. On the other hand, in AI programs since each knowledge structure has a concrete boundary, it is easier to develop and handle them. Such portability is important when we construct a practical NLP program. Our strategies are the followings:

- **To place right paradigms in the right places:**
  Our goal is to create a practical NLP model which handles **atmosphere** as well as other required features for NLP. Since ANN and AI have many contradictory characteristics, it is difficult to construct a unique NLP paradigm which contains all advantages of both approaches. Therefore, the model will be a cooperative model of AI and something which holds good features of ANN enough to handle **atmosphere**.

- **To extract good features of ANN and to create simpler mechanisms:**
  Presently ANN models can not be practical NLP programs because they still have many functional problems and development difficulties. However, it is possible to skim the needed features for expected functions from the ANN model and to create simpler mechanisms for them.

- **To add good features of ANN without destroying AI's knowledge structures:**
  Although ANN still adopts knowledge concepts like schema or script which AI once proposed, ANN destroys AI's conventional structural skeletons for such knowledge concepts [14]. Instead, ANN introduces microfeature based distributed representations. Our strategy is to

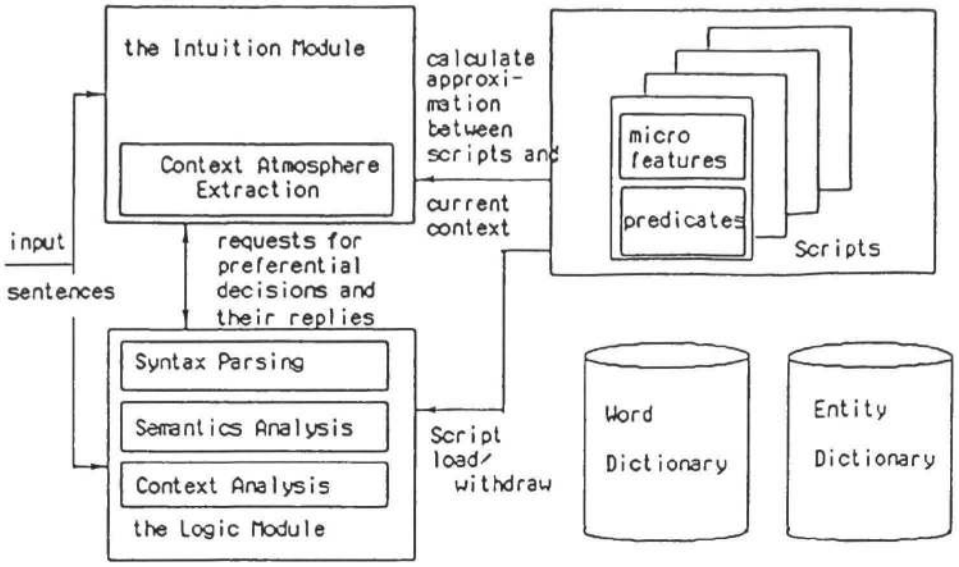

Figure 1: Overview of the Cooperative Model

keep the AI's various structural skeletons for such knowledge concepts and to stuff ANN based representations into such structural skeletons.

OVERVIEW OF THE COOPERATIVE MODEL

The cooperative model consists of two separate modules; a logic module and an intuition module. The logic module and the intuition module run in parallel. Both modules get the same input sentences. Figure 1 shows the overview of the cooperative model. The logic module does ordinary syntax parsing, semantic analysis and context analysis to input sentences. In addition to these, it checks for the justifications of various preferential decisions done by the intuition module. The intuition module extracts a current context atmosphere as analyzing input sentences word by word basis. By using the current context atmosphere, the intuition module makes preferential decisions according to the demands of the logic module like the preference of semantics for ambiguous words and the preference of memory retrieval, for example, like selecting appropriate *Ron* among many Rons. Further, the intuition module monitors scripts and notifies the logic module of the proper selection/withdrawal of scripts.

Each knowledge structure holds two different representations; predicate style knowledge for the logic module and microfeature style knowledge for the intuition module. Microfeature definitions are placed per each word definition, entity knowledge, and script. This dual definition has an advantage from the implementation point of view. Since each knowledge structure has its concrete skeleton, its development, modularity, maintenance, and partial change are easy, comparing with fully distributed representation like [14].

The Intuition Module

The intuition module keeps ANN's good features and is constructed by much simpler mechanisms. It is characterized by **microfeature representation**, **chronological FIFO**, and **threshold-based script selection**.

Microfeature representation [11] is well-suited to express **atmosphere** of a concept. **Atmosphere** at a specific situation is defined as a collection of **atmospheres** of words appearing lately in input sentences. For example, if there appear words like *a written oath*, *candle*, *ring* and *march*, an atmosphere like *as for marriage* will be suggested.

A simple **chronological FIFO** structure can be the container to store microfeature expressions of words appearing lately. Microfeature expressions of lately appearing words are pushed into the FIFO and each microfeature is calculated its number of appearances in the FIFO. The set of microfeatures whose number of appearances is more than the system defined threshold becomes the microfeature expression of **atmosphere** for the current situation. Microfeatures which become old in the FIFO are automatically abandoned. However, if a context proceeds in a same topic, the microfeature organization in the FIFO does not change rapidly because words appearing in a same topic must hold similar microfeature expressions.

The preferential decisions for lexical disambiguations or right memory retrievals are done by the comparison of distances between the microfeature expression of the current situation and microfeature definitions of candidate concepts or memories. For example, if a word has two different meanings, a meaning whose microfeature expression is closer to that of the current situation is chosen.

The selections of scripts are done in the similar manner. Each script is defined its characteristics with a microfeature expression. If the distance between the microfeature expression for a specific script and that of a current situation becomes smaller than the system defined threshold value, the script is regarded as a proper script for the current situation. If a script is selected, the script changes the microfeature organization in the FIFO. Each script is beforehand enumerated several keywords in it to express the characteristics of the script. Each keyword has its microfeature expression. If a script is chosen as a proper script, the microfeature expressions of enumerated keywords in the script are added into the FIFO. As the effect, many older microfeatures in the FIFO are chronologically abandoned and the contents in the FIFO become dominated by the newly entering microfeatures.

In AI based NLP programs, each typical situation is discretely defined like script. In order to make the discretions fine, various approaches have been proposed like a discrimination-net in FRUMP [4] or a hierarchical tree in ATRANS [12]. However, there sometimes happen situations which should be located between classifications. The **threshold-based script selection** approach achieves more continuous discrimination and selection of scripts than such conventional approaches. Instead of selecting only one fit script, it regards all scripts whose approximation to the current context are close enough as proper scripts, and adopts them.

The Logic Module

Since the logic module must find logical mistakes in the preferential decisions of the intuition module, justification mechanisms are necessary for the logic module. Therefore, Truth Maintenance System (TMS) [6] was introduced for the logic module. In TMS each predicate is followed by its justifications which support the predicate. If a conflict occurs between two predicates, TMS traces back along the justification links (dependency-directed backtracking) and discovers the causal predicate of the conflict. We add a new justification type, *by-preference* in TMS. This justification type is the weakest compared with other types of justifications. For example:

S2. Person: How is Ron going?

S3. System: He will be divorced next week.

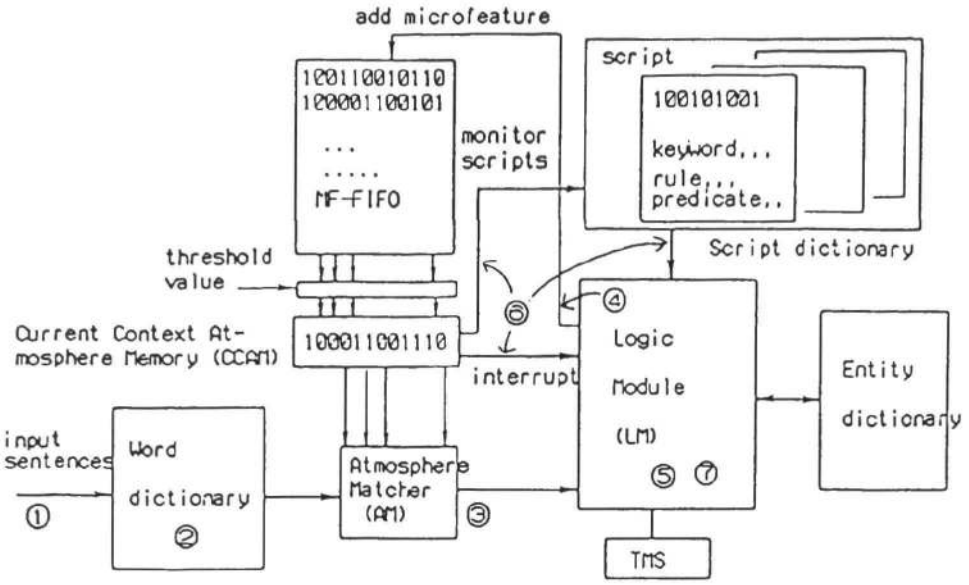

Figure 2: Configuration of the Cooperative Model

S4. Person: No, he is single.

After reading S2, the system searches its memory space and finds several memory entities each of which expresses a person named "Ron". Then, it calculates the distance between the microfeature expression of the current situation and the microfeature definition of each candidate memory entity. The system selects the most approximate entity to the current situation among them and regards it as *Ron* mentioned in S2. Then, it replys S3 to the person. After reading S4, TMS in the logic module of the system finds a logical conflict since the assumed *Ron* is not single. Then, TMS discovers a predicate which says that the memory entity for *Ron* was selected as *by-preference*. The predicate is removed and the preference is retried and a new entity for "Ron" is chosen again.

DETAILS OF THE SYSTEM

System Configuration

Figure 2 indicates the whole system configuration. The system consists of the following modules:

- Microfeature FIFO (MF-FIFO) and Current Context Atmosphere Memory (CCAM);
- Atmosphere Matcher (AM);
- Logic Module (LM), which is monitored by TMS;
- Script dictionary, Word dictionary, and Entity dictionary.

Word dictionary holds the meanings of words and phrases. Entity dictionary holds the memories of various entities like many Rons this system memorizes. Script dictionary holds scripts. Each word, entity and script has two different representations; microfeature representation and predicate representation. The microfeature representation is implemented as a fixed-sized bit vector. The

following is the general form of a script in Script dictionary.

Script *script-name:*
Microfeature definition,
Keyword, keyword, keyword,...
Local rules available in this script,
Local predicates available in this script.

Each script holds several keywords which characterize the script, local rules which analyze sentences under the local domain of this script, and local predicates which describes the local domain of this script.

MF-FIFO, CCAM and AM construct the intuition module. MF-FIFO is a FIFO whose lengths are several dozens and whose widths are the same as that of microfeature bit vectors. CCAM calculates the number of ON (1) bit per microfeature in MF-FIFO, then selects microfeatures whose number of appearances is greater than the system defined threshold. The set of the selected microfeatures is also expressed in the form of a bit vector and is used as the current context atmosphere. The microfeature set in CCAM changes when a new input microfeature is added into MF-FIFO. AM gets an input in the form of microfeatures, then calculates the distance between the input and the state of CCAM.

LM is monitored by TMS. Justification links of TMS are made from predicates in LM to other predicates in LM. A justification is also linked from a predicate in LM to a bit vector in MF-FIFO if the bit vector was pushed into MF-FIFO by the predicate. It means that contents in MF-FIFO are also monitored by TMS.

Flow of the System

The typical flow of this model is described in the following sequence. Numbers from (1) to (7) in Figure-2 are corresponding to the following processes.

(1) A new input sentence is given to the system.

(2) Lexical meanings for each word in the sentence are taken from Word dictionary. Each meaning consists of two different representations; the predicate form of logical definitions and the bit vector form of microfeatures. If a word has two or more meanings, all lexical meanings are taken.

(3) AM receives the lexical meaning definitions. If there is only one meaning for a word, the meaning definition is soon passed to LM. If a word has two or more meanings, AM compares the microfeature definition of each meaning with that of CCAM. According to the degree of the approximation between each meaning and CCAM, preferences are given among the competing meanings. Then, all of them are passed to LM.

(4) LM employs the most approximate meaning to the current context atmosphere among competing meanings. The logical definitions of the employed meaning are loaded into LM in the form of predicates, and the microfeatures of the employed meaning are pushed into MF-FIFO in the form of bit vector. The bit vector in MF-FIFO is pointed from the corresponding predicate in LM by a dependency-directed link of TMS.

(5) Syntax parsing, logical semantic analysis and context analysis are done in LM. When a predicate is newly derived by logical inferences in LM and if the predicate has its microfeature definition, the microfeature definition is added into MF-FIFO.

(6) The system always measures the distance between CCAM and each script in Script dictionary. If a distance between CCAM and a script becomes smaller than the system defined threshold, CCAM

asynchronously sends an interrupt to LM and notifies that the script is proper to be selected as the current context. Local rules and local predicates of the script are taken from Script dictionary and loaded into LM. Keywords of the script are added into MF-FIFO in the form of microfeatures. If a distance between CCAM and a script which is currently loaded into LM becomes larger than the system defined threshold, CCAM asynchronously notifies the event to LM. Then, LM removes the local rules and the local predicates of the script from LM. If there exist bit vectors in MF-FIFO which were justified by the removed predicates, such bit vectors are also removed from MF-FIFO at the same time.

(7) If a contradiction occurs among logical predicates in LM, TMS finds its causal predicate by tracing back dependency links, then removes the causal predicate. If the causal predicate was justified by *by-preference*, LM employs another candidate according to the preferential orderings.

RELATED WORKS

Hendler [10] combines microfeature with the marker passing approach. While he unites microfeature and marker passing, we constructed a cooperative model of microfeature based module and logic based module. Wilks's preferential semantics [18] is similar to our intuition module. But his model does not have a logical justification mechanism for preferential decisions. Charniak [2] proposes the cooperation model of a marker passing module and a logic module. He relies on the marker passing mechanism and lets the mechanism do as many things as possible including higher level inferences. Our stance is the opposite. We regard the logic module is main and the intuition module is its assistant since the preferential decisions of the intuition module is doubtful. Charniak and Goldman [3] propose another model which is based on logic. They introduce ATMS [5] and use its justification mechanism. ATMS generates many assumptive interpretations. However, it is not sure how their model makes preferential decisions among such many interpretations.

CONCLUSIONS

In this paper, we have presented a *solving by intuition* method. We introduced **atmosphere** as an instance of *solving by intuition*. **Atmosphere** is useful for right memory retrievals, preferential orderings, and script selection/withdrawal in NLP. **Atmosphere** is realized by a microfeature based mechanism which was inspired by ANN researches. The mechanism can be seen as a simplified implementation of ANN. It is characterized by microfeature representation, chronological FIFO, and threshold-based script selection. The microfeature based mechanism works for NLP as an intuition module cooperating with a logic module. The logic module does syntax parsing, semantic analysis and context analysis. In addition to these, it checks for the justifications of preferential decisions done by the intuition module. The cooperative model can construct a practical NLP program since it is easy to develop, modularize, maintain, and change knowledge structures in the model.

ACKNOWLEDGEMENTS

The authors would like to express their appreciation for continuous encouragement from Kazumoto Iinuma and Takashi Araseki. One of the authors stayed at the Artificial Intelligence Laboratory, UCLA for the 1987-1988 academic year. He would like to thank Prof. Michael G. Dyer for giving him the opportunity to work at UCLA, and for his technical suggestions throughout the work. He thanks Ron Sumida for his technical assistance in NLP and ANN researches. He'd also like to thank other lab members. They thank to Shinji Yanagida for his support to install LaTeX.

References

[1] Charniak, E.C., "Passing markers: A theory of contextual influence in language comprehension.", Cognitive Science 7(3), July Sep, 1983.

[2] Charniak, E.C., "A Neat Theory of Marker Passing", AAAI-86, 1986.

[3] Charniak, E.C. and Goldman, R., "A Logic for Semantic Interpretation", ACL, 1988.

[4] DeJong, G., "An Overview of the FRUMP system", in "Strategies for Natural Language Processing", Erlbaum, 1982.

[5] De Kleer J., "An Assumption-based TMS", Artificial Intelligence, Vol. 28, No. 2, 1986.

[6] Doyle, J., "A glimpse of Truth Maintenance", in "Artificial Intelligence: An MIT Perspective", MIT Press, 1979.

[7] Dyer, M.G., personal communications in his class at UCLA, 1988.

[8] Dyer, M.G., Flowers, M. and Wang, Y.A., "Weight Matrix = Pattern of Activation: Encoding Semantic Networks as Distributed Representations in DUAL, a PDP Architecture", Tech. Report UCLA-AI-88-5, AI Lab., UCLA, 1988.

[9] Feldman J.A. and Ballard, D.H., "Connectionist models and their properties", Cognitive Science, 6, 1982.

[10] Hendler, J., "Marker-passing and Microfeatures", IJCAI-87, 1987.

[11] Hinton, G.E., McClelland, J.L., and Rumelhart, D.E., "Distributed Representations", in "Parallel Distributed Computing Vol. 1", the MIT Press, 1986.

[12] Lytinen, S.L. and Gershman, A., "ATRANS: Automatic Processing of Money Transfer Message", AAAI-86, 1986.

[13] McClelland, J.L. and Kawamoto, A.H., "Mechanisms of Sentence Processing : Assigning Roles to Constituents of Sentences", in "Parallel Distributed Processing. Vol. 2", the MIT Press, 1986.

[14] Rumelhart, D.E., Smolensky, P., McClelland, J.L. and Hinton,G.E., "Schemata and Sequential Thought Processes in PDP Models", in "Parallel Distributed Processing. Vol. 2", the MIT Press, 1986.

[15] Rumelhart, D.E. and McClelland, J.L., "Parallel Distributed Processing Vol. 1 and 2", the MIT Press, 1986.

[16] Schank R.C. and Abelson, R. "Script, plans, goals and understanding", Erlbaum, 1977.

[17] Waltz, D. and Pollack J., "Massively Parallel Parsing", Cognitive Science, 9, 1985.

[18] Wilks Y., "An Intelligent Analyzer and Understander of English", CACM, Vol. 18, No. 5, 1975.