NATURAL LANGUAGE PROCESSOR IN DILOS SYSTEM

Gregory Senin

January 1978 WP-78-2

Working Papers are internal publications intended for circulation within the Institute only. Opinions or views contained herein are solely those of the author.

2361
Laxenburg
Austria

International Institute for Applied Systems Analysis

Review

Author: G. Senin

Title: Natural Language Processor in DILOS System

This paper outlines some of the philosophy and describes the structure of the Natural Language Processor in DILOS. It is rather brief and does not really discuss in any depth the reasons for the basic structure chosen or relate it to other attempts of natural language processing. As an introduction it is quite adequate for this specific system and should probably be a working paper. The paper has been edited by this reviewer to improve its consistency and clarity of usage.

Abstract

The paper describes underlying ideas and operation of the natural language processor, which is a part of the Dialog Information Logical System (DILOS), (Computer Center of the USSR Academy of Sciences, Moscow).

The natural language comprehension is assumed to be performed within:

(a) general context, determined by the whole system purpose;

(b) some local context, connected with current data base.

The basic parts pf the processor are described: the main program, transition network and current vocabulary as well as special mechanisms provided for tackling homonymy and words, unknown to the system. Some suggestions are proposed for combining the existing system with a system with a syntactical analyzer.

I. INTRODUCTION

In this paper the underlying ideas and operation of a natural language (NL) processor ("SPEAK"), which constitutes a part of the DILOS system are described.

The system was developed in Moscow (Computer Center of the USSR Academy of Sciences) and partly transferred to the IIASA PDP 11/45 computer [1, 2]. The whole system is written in LISP, which although makes it less efficient, provides the advantage of machine-independent design, transparency and portability.

II. FRAMEWORK

Any NL communication occurs in a definite environemnt (context). This is important to underline, especially with regard to communication with an artificial system (program). To feed a computer that has absolutely no human experience with a great deal of human knowledge seems rather difficult for two reasons:

1. The capacity of today's computers is comparatively low;

2. So far we are not able to adequately organize our knowledge to put it into a computer.

Therefore, we prefer to restrict the scope of our considerations each time we have difficulties either with the volume or organization our data. The approach described here avoids or at least diminishes some problems peculiar to the NL phenomena:

- Wide lexicon: some words introduced by the user are likely to be "unknown" to the system;
- Homonymy: some words have more than one meaning;
- Rather complicated syntax and its indirect correlation with semantics: even if we obtain syntactical structure, it is not straightforwardly transformable into semantic form [3].

The processor has been developed bearing in mind the following application environment [4]:

- Pragmatic or Operational, basis of the communication: each input phrase tends to be converted into a command for some operation on data. This feature is predetermined by the purposes and capabilities of the whole system and constitutes what we call general context of our communication;
- Problem Area Orientation, "at every moment" we deal with a particular piece of information, rather homogeneous and rather independent of the remaining part (local context).

III. CONFIGURATION

The operation context suggests the two-level process comprehension and execution. Comprehension proper (that is performed by SPEAK) results in generating a formal expression in some operational language (ϕ-language). This ϕ-expression then is passed to an executive part of the system (execution).

In our version ϕ-language is a language for data base management (information retrieval and amendment). The scope of operations embraces a data base (DB) which embodies current local context. As a rule such a DB represents a model of some problem area structured accordingly to the ϕ-language syntax.

A. The Three Parts of SPEAK

SPEAK includes three separate parts:

- Main program (MP), which contains basic mechanisms independent of a particular environment of the system;
- Current vocabulary (VOC);
- A finite state automaton, or transition network (ATN);

i) The Main Program (MP)

- Extracts out of the VOC information corresponding to the input phrase lexicon;
- Works up (through the ATN) current word;

- Interprets homonymic and unknown words;
- Builds up ϕ-expression in appropriate points and provides termination of the analysis.

To estimate the role of vocabulary it should be kept in mind that each VOC virtually links user's lexicon ("terminology") with the DB contents.

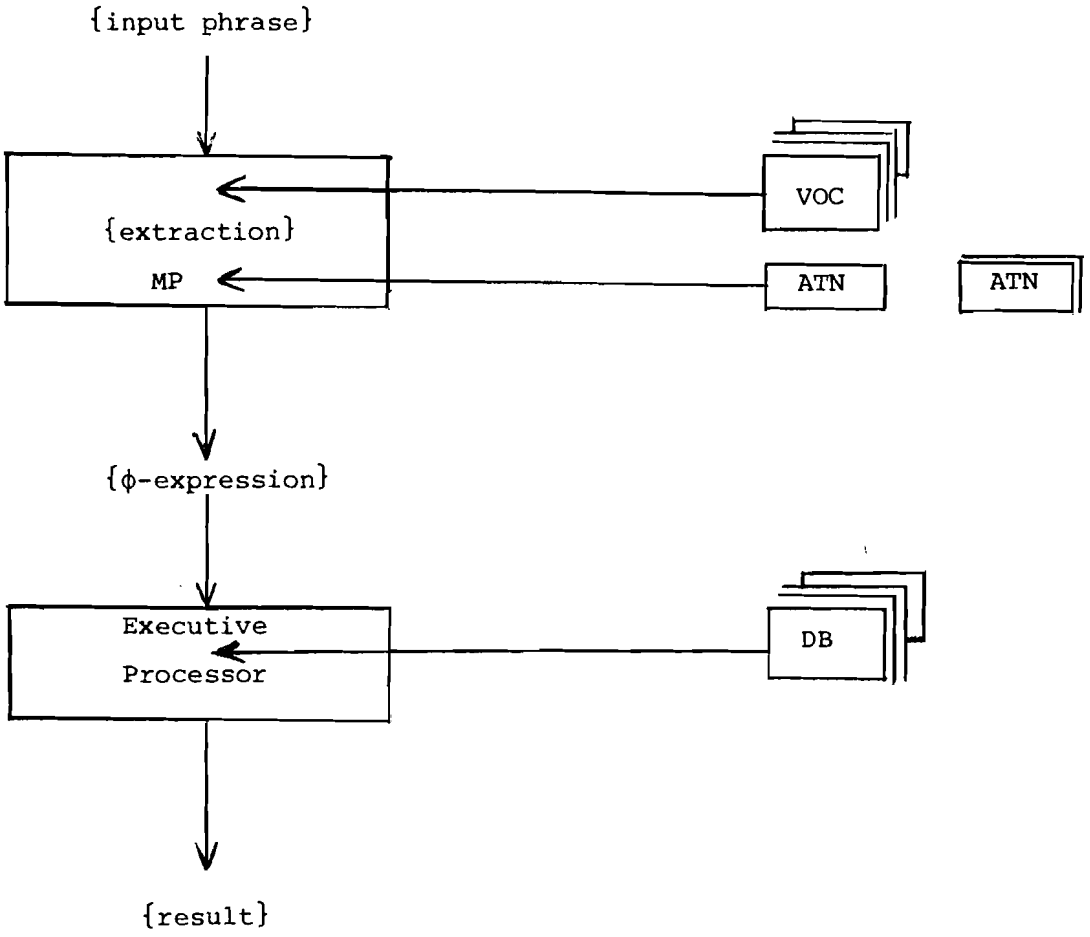
Thus, generally speaking, each DB (or, more precisely, each {DB, user} pair) generates its own VOC and the same word in different VOCs may have quite different meanings.

It obviously facilitates description of meanings and dimishes homonymy, but also creates the difficulty of adjusting the system to particular DB context.

The ATN is a structured set of programs, one of which becomes associated with the current word of an input phrase. The whole body of these programs is directed to generate "regular" ϕ-expressions. Thus, the ATN is a function of ϕ-syntax and can be regarded as a physical embodiment of the general context.

To modify ϕ-language (and formal representation of data), we have to replace ATN contents with another, without changing its structure (or if it is possible to make interface translator $\phi_1 \rightarrow \phi_2$).

Figure 1: General Scheme of the Whole Process



We can write it in functional form as follows:

SPEAK = F (voc, atn), where in turn

VOC = f_1 (DB, NL, user) and ATN = f_2 (ϕL).

ii) Vocabulary Structure (VOC)

Some preliminary remarks about ϕ-language. Any formal language of this sort can be described in some metalanguage terms.

Consider a simple example of retrieval ϕ-language.

< ϕ-expression > → < operation > < division > < description >: < prescription >

< description > → ϕ | < property > / < pattern >/

< prescription > → ϕ | < property >

Let s-types be the atomic non-terminal constituents of this metalanguage, e.g., {operation, division, property, pattern}.

In real expressions they are substituted by terminal values that we call s-codes (or codes), e.g.:

< operation > FIND | DELETE | APPEND ...

< division > SCIENTISTS| ... ENERGY| ..... KINO|...

< property > AGE ... LOCATION ... DURATION .....|1|2 ..

An example of ϕ-expression could be:

FIND SCIENTISTS AGE /40/: NAME

with the obvious "meaning":

"find among the scientists everyone who is 40 years old and obtain their names".

We can notice that some words here correspond to (s-type, s-code) pairs: old ↔ (property, AGE), 40 ↔ (pattern, 40). We can suppose that other words (if of any importance) rather refer to NL pecularities and pay auxiliary roles in the construction of ϕ-expressions, not necessarily occurring in them. These roles can also be classified. Thus s-types emerge from the ϕ-syntax and partly from some "linguistic-heuristic" considerations.

Each record in a VOC associates a particular word W with definite s-type Sw and (possibly) with corresponding code Cw:

W ↔ Sw, Cw; or (W(tp Sw cd Cw)) in LISP

Thus, some words are regarded as candidates for filling vacancies in the constructed ϕ-expression.

Generally speaking, a one-to-one correspondence does not exist between words and s-types (although approximately there is)

So we admit two other cases:

(a) Composites, i.e. words with "more than atomic" sense (i.e. type); such words are assigned a sequence of atomic s-types in the form of LISP list: (S1 ... Sn), that means "S1 + ... + Sn".
For example, "older ..." = "age+beyond ..."
"woman" = "person+sex+female"
(if the words in the right parts are "atomic")

(b) Homonyms, i.e. words with several meanings, obtain as s-type an "alternative" list: (, S1 ... Sn), that means "S1 or ... or Sn" (each Si may be either atomic or composite).

MP supplies each word from the input phrase with the VOCabulary information and passes to the next stage.

iii) ATN Structure

The ATN is a set of records, each of those represents a "state" of the automaton. Each state contains a prediction (in the form of list) of likely s-types, "expected" in the state.

Further, with each (state, s-type) pair a specific program (PROG) is connected.

Normal actions in a PROG are:

- Building up some piece of ϕ-expression;
- Changing contents of some important variables ("registers"), that influences further analysis; in particular, changing the state of the ATN.

Each PROG also returns a value that indicates to some point in MP, from which the analysis goes on. The most "usual" points of return are "jump" (that means "proceed with the next word") and "move" (means "proceed with the current one"). Other two points serve:

"upset" - for tackling homonyms and unknown words (see below);
"finis" - for building up the resulting ϕ expression and termination of the analysis.

Thus, in fact the PROG associated with each word is a function of the word s-type and the ATN state.

These programs are evaluated one-by-one while MP passes along the input string. Since ATN is a parameter of SPEAK, it can be augmented by recursion - with subnetworks processing ϕ-language substrings (that may posses their own detailed syntax).

B. Processing of Composites, Homonyms and Unknowns

A composite, being encountered, suspends the normal word-by-word analysis. Atomic elements, listed in the composite type, are processed one-by-one until the list is exhausted, after that the normal process resumes. When dealing with a homonym, MP creates a branch point (BP). In each BP the content of necessary variables is stored to have the possibility of recovery if further analysis fails (i.e. an ATN-PROG returns "upset"). When this happens, MP passes on to the next alternative type Si. If all alternatives in the given BP comes back to the previous BP, thus covering the entire tree of alternative paths.

Any word in a VOC may be marked as "nil", i.e. "unimportant" and then it is simply ignored while scanning. However, when looking through the VOC, MP can discover some words, not present in it and still employed by the user. During the first scanning MP omits them, but if necessary it is also able to make some predictions about their possible meaning. Namely, each unknown word also creates BP, but not earlier than all paths produced by homonymic words fail. Unlike homonyms, unknown words acquire the alternative type not from the VOC, but as a rule from the prediction list, drawn out of the current ATN state. The content of this list may be varied according to the system adjustment to particular NL, DB and so forth.

IV. TAKING INTO ACCOUNT NL SYNTAX

It may seem surprising but for the time being SPEAK does not need any syntactical analysis (SA). Of course, it proves only that the areas chosen for application have been rather narrow and simple. However, syntactical considerations (if any) could be combined rather naturally with the present system. So far, a string of input phrase words supplied by vocabulary information serves as an immediate in-

put of the ATN process. But this string can be considered as a result of some preliminary processing such as syntactical.

Further, any syntactical tree can be represented in a list form, with parenthesis serving as special delimiters. Moreover, the result of SA is not necessarily intended to be a tree, but merely a set of "small" trees not linked together. At last, we can consider a "weak syntactical analysis" according to the following: the effect it is required to produce is reordering of initial string of words into a "more regular" form taking into account some syntactical reasons. And naturally we shift crucial considerations to the ϕ-oriented stage of analysis.

V. CONCLUSION

We have performed first testing of the system in Moscow and at IIASA and assess it as hopeful. Not dealing with NL syntax, the SPEAK processor admits as input almost all range of expressions intermediate between NL and ϕ-language.

Probably it would be psychological better for users to start with more formal language and then gradually move to a "more natural" one thus getting accustomed to the system. In fact, it means establishing real context of communication very much like in a human-to-human case.

Possible improvements of the SPEAK operation could concern:

- Combining it with a syntactical processor (likely, a "weak" one);
- Capability of efficient dialog with user as a way of better understanding;

We plan to introduce necessary modifications during the next two years.

References

[1] Briabrin, V. M. "DILOS Reference Manual: Part I", IIASA, RM-76-52, July 1976.

[2] Briabrin, V. M. "Natural Language Access to the Model Data Base", IIASA (RM), to be published.

[3] Briabrin, V. M., and G. V. Senin, "Natural Language Processing Within a Restricted Context", International Workshop on Natural Language for Interaction with Data Bases, IIASA, January 1977.

[4] Schank, R. et al. "Conceptual Information Processing", North-Holland Publishing Co., Amsterdam, 1975.