Principles and Techniques of Natural Language Parsing : A Tutorial

Keh-Yih Su

蘇克毅

National Tsing Hua University

國立清華大學

Proceedings of ROCLING I(1988)
R.O.C. Computational Linguistics Workshops I pp 55-61

中華民國第一屆計算語言學研討會論文集 55-61 頁

Principles and Techniques of Natural Language Parsing : A Tutorial

Keh-Yih Su

Department of Electrical Engineering

National Tsing-Hua University

Hsinchu, Taiwan, R.O.C.

Abstract

To be able to parse sentences successfully is a preliminary requirement of a natural language processing system. Therefore, parser is a very important part of a natural language processing system. This paper will discuss the typical procedure of natural language processing; examine the difference between the natural and the artificial languages; list the criterions for designing a natural language parser; and take a look at various parsing strategies along with some commonly used parsing algorithms employed in natural language processing system. Finally, some helpful tips on building a natural language processing system will be provided.

Introduction

After the computer was invented, people soon discover that natural language processing would be an important field for the non-numerical application of computer. In a natural language processing system, to parse sentences successfully is very important. Therefore, this paper will address this problem in the following sections. In section 2, the typical procedure and problem of processing the natural language will be presented. In section 3, we will take a look at the differences between artificial language and natural language. In section 4, several criterions are listed for designing a parser for doing natural language processing. In section 5 and 6, we will survey various parsing strategies and parsing algorithms available (they will not be discussed in detail because of the length limitation of this paper). Lastly, we will give some helpful tips for building a natural language processing system based on the experience we had gained from our on-going project on an English to Chinese machine translation system.

Before we discuss the issues of parser in a natural language processing system, we would like to define some terms first in the context of the natural language processing to familiarize the reader with the field. They are languages, researchers and natural language processing.

Languages can be classified as : natural languages, artificial languages, sublanguages and controlled languages. Natural languages are languages like Chinese and English. Artificial languages are languages used in programming like Fortran, Basic and C. Sublanguages are subsets of a language abstracted from a restricted domain (e.g. Computer Science or Linguistics). Controlled languages, which can be further divided into weakly controlled and strongly controlled, are languages that are intended to restrict the speaker to use them in a rigid form in order to decrease the degree of handling difficulty.

On the other hand, researchers in natural language processing can be classified into three types according to their research interest. They are theoretical linguists who are interested in linguistic phenomena in general; computational linguists who are more application oriented and interested in domain or language specific linguistic phenomena; and language engineers who are interest in providing friendly working environment to support the work of the computational linguists [NAGA 88].

As for the natural language processing, it can be separated into three types. The first is the analysis process which takes in strings in natural language and turns them into logic semantic form. The second is the generation process which takes logic semantic form and turns it into natural language strings. The third is the understanding process of a natural language which would take in strings in natural language and turn them into logic semantic form and then it is capable to answer questions in natural language form [ALLE 87][GROS 86].

The Typical Procedure of Natural Language Processing

A typical procedure of transforming sounds into meaningful logic semantic form in natural language processing can be separated into phonetic and phonological analysis, morphological analysis, lexical analysis, syntactic analysis, semantic analysis and discourse analysis. In page 17 of [WINO 83], the connecting relationship between analysis phases is clearly depicted along with the input and output requirements of every phase.

These analysis phases can be combined into several passes in different ways. For example, in a one pass translator, lexical analysis, syntactic analysis, semantic analysis are combined along with the code generator into one single pass (syntax directed translation). In this case, the source program is traversed once and resulting code is generated immediately. As another approach, we can separate these analysis phases into different passes by incorporating one or more phases into one pass.

Natural language processing is a complex task. The most troubling problems in the task are the ambiguity and the ill-formedness of the sentences. Ambiguity can be divided into lexical ambiguity (multiple categories), syntactic ambiguity (several possible structures for a given sentence), semantic ambiguity and pragmatic ambiguity [HIRS 87][GODB 82] . Some examples are given below :

1. Lexical Ambiguity : *Design* Can be either a noun or a verb.
2. Syntactic Ambiguity : *He saw a man with a telescope.*
   a. He saw the man through a telescope (with 'a telescope' modifying the verb 'saw'), or
   b. He saw a man who had a telescope with him (with 'a telescope' modifying the noun 'man').
3. Semantic Ambiguity : *Police were ordered to stop drinking by midnight.*
   a. Every policeman must not drink (implying liquor) after midnight, or
   b. Police must stop people from drinking beyond midnight.
4. Pragmatic Ambiguity : *It is cold in here.*
   a. If the speaker is in a very cold room, this sentence might mean that he wishes the heater be turned higher. (indirect speech)
   b. If the speaker is under a hot sun, he might mean that he wants to enter the cool room. (direct speech)

The other trouble spot is the ill-formed sentences. This can also be subdivided into lexical, syntactic, semantic and pragmatic ill-formedness [GODB 82][NIRE 87]. some examples of these are :

1. Lexical Ill-formedness : Typing error recoverable by human but not by computer.
2. Syntactic Ill-formedness : *Which Seven ?* Which is ungrammatical but still understandable by human who read the sentences before this one.
3. Semantic Ill-formedness : *A red apple is walking on the street.* This sentence is semantically ill-formed because apple does not walk. But, a human would accept it if this sentence is from a children's book.
4. Pragmatic Ill-formedness : *How are you? It is very sunny outside.* The conversation is pragmatically ill-formed because the answer does not match with that of the question.

The Difference Between Artificial Language and Natural Language

Because the parser commonly used for natural language processing system is usually evolved from the parser for artificial languages, therefore we must take a look at the differences between natural language and artificial language in order to know why and how the parser for the artificial language must be changed. The main difference between artificial languages and natural languages is that artificial language has small vocabulary, and usually has a simple context-free grammar, and does not change often. In addition, artificial language does not need any morphological analysis and does not have any ambiguity in syntactic category, parse tree or semantic interpretation. Also, every accepted sentence in artificial language can be parsed deterministically and never is ill-formed. These sentences also are rarely context dependent so in general they do not need discourse analysis. Because of all these reasons, if we want to utilize the parser designed for programming language to do the parsing for sentences in natural language, it must be augmented first.

Criterions for Designing a Natural Language Parser

There are several criterions that need to be carefully considered in designing a natural language parser. The first criterion is the cost of the designing, which can be lowered by using well-established theories and tools currently available.

The second criterion is the execution efficiency. There are several ways for improving the runtime efficiency. One is to use parsing algorithm that has small branching factor, which means there will be less paths to try and thus less states that need to be checked. If wrong paths are abandoned as early as possible, the runtime efficiency will also be improved. But how early will depend on the lookahead capability of the parsing algorithm. It is also better to employ chart parsing concept [WINO 83] and the idea of having several logical paths sharing a physical path [TOMI 87] to avoid redundant parsing. Another efficiency-improving method is to have several logical subtrees sharing the same physical subtree to avoid unnecessary copying of subtrees. And lastly, score can be used to do predictive state space search and thus reduce the size of the search space.

The third criterion is to have a small memory space requirement. This can be done by choosing a suitable parsing algorithm and by sharing the physical structures of subtrees. In general, a bottom-up parsing algorithm will require large memory space, and the more lookahead there is the more memory will be required. As for sharing of subtree structures, the basic idea behind it is to remove the empty entries and compact the data storage area.

The fourth criterion is the power of the parser. There are several factors that require close examination under this criterion.

- Linguistic felicity: whether the linguists can describe their knowledge with only a few statements.
- Expressiveness: what class of knowledge can be incorporated into the system.
- Handling of Ambiguity: whether the parser can select the most likely one.
- Handling of inexact knowledge: whether the uncertainty can be added into rules.
- Handling of ill-formed sentence: whether such construct can be elegantly handled via Fail-Soft or some sort of Error Recovery (According to the report of Eastman and McLean, 1981, - 26.7% of English queries are ill-formed [EAST 81]).
- Error diagnostic: the ability to pinpoint the error and the reason of the error to speed up the debugging time.

The fifth criterion is the portability. This issue includes (i) porting of parser between different computer systems, like from PC to SUN micros; (ii) or changing of the rules in the underlying grammar; (iii) or changing of the source or target languages in the system (e.g. translating of English to Chinese is changed into from Japanese to English); (iv) or changing of the grammar formalism used as the system's underlying grammar (e.g. from GPSP to LFG). The problem of portability is also closely dependent on what the system architecture is, which programming language is employed and how modular the system is.

The sixth criterion is the maintainability. This means that the system should be modular; modifications to knowledge base should be incremental; documentations should be kept and a good programming style should be strongly enforced in order to achieve easy maintainability.

Above are some of the basic criterions for designing a natural language parser. In an actual implementation, the requirement for these criterions will vary for different applications. Therefore, there will be compromise between these criterions.

Various Parsing Strategies

Parsing strategies can be classified into different categories according to different points of view. Due to the length limit of this paper, we will just list them along with references.

Parsing strategies based on different driving mechanisms : 1. Template Matching [NIRE 87], 2. Lexicon Oriented [NIRE 87], 3. Syntax Oriented [NIRE 87], 4. Semantic Oriented [WINS 84] and 5. Mixed [NIRE 87].

Parsing strategies based on syntax tree constructions : 1. Top-Down (Expectation driven, or Hypothesis driven) [AHO 72], 2. Bottom-Up (Data driven) [AHO 72] and 3. Mixed (usually uses subgrammar) [MARC 80][SU 87].

Parsing strategies based on path traversal : 1. Backtracking [AHO 72], 2. Parallel (requires large memory space) [AHO 72] and 3. Mixed (parallel with smaller subtrees) [YONZ 88].

Parsing strategies based on goal state searching algorithm : 1. basic path searching algorithm [WINS 84] : a) Depth-First, b) Breadth-First, c) Hill-Climbing, d) Best-First and e) Beam **2. Optimal path searching algorithm [WINS 84] :** a) British Museum, b) Branch and Bound, c) Dynamic Programming and d) A*.

Parsing strategies based on branching [AHO 72] : 1. Deterministic Parsing (with lookahead buffer) and 2. Nondeterministic Parsing.

For an unambiguous grammar, a nondeterministic parser can be converted into a deterministic parser if the look ahead is far enough.

Parsing strategies based on different input sentence scanning [WINS 84] : 1. Left to Right, 2. Right to Left and 3. Middle out.

Which is better will depend on the predictability of the sentence in the scanning direction. The one with less entropy is the better one to use.

Parsing strategies that avoids redundant parsing : 1. Chart Parsing [WINO 83] and 2. Merge Logic Path [TOMI 87].

Some Well-Known Parsing Algorithm

Some of the well-known parsing algorithm used in natural language processing are : 1. LR [AHO 72], 2. Left Corner [AHO 72], 3. ATN [WINO 83], 4. C.Y.K. [AHO 72], 5. Earley [AHO 72], 6. Marcus (lookahead buffer & rule packet) [MARC 80] and 7. Tomita [TOMI 87].

Some Tips in Building up a Natural Language Processing System

After surveying the basic issues in natural language processing system, following are some tips for building a natural language processing system. These tips are based on the experience we had gained from our machine translation project.

- First, if you do not have the experience of building up a similar system before, try to build a small pilot system to begin with in order to gain the essential hands-on experiences.
- Second, since this is an interdiscipline research, communication problems may arise between the linguist and the computer/software engineers.
- Third, because while building a large system, it is likely to become an engineering problem, so the guidelines from the software engineer should be closely followed.
- Fourth, it is better to do what you think is proper instead of rigidly following the theory in the books. You should also take precautions in employing theories that are still under debate.
- Fifth, execution efficiency and portability of the system is very important.
- Sixth, when tradeoffs are being weighted between quality, speed and memory space, you should foresee the progress in the computer systems.
- Seventh, you should maintain a test file for doing "regression test" and "time test" when modifying your system.

- Eighth, the development of both system and the environment (or tools) should be done at the same time.

Conclusions

In a natural language processing systems, parser plays an important role and has a large influence on the performance of the system. Although research in parsing mechanism has a very long and fruitful history, it is still an active research domain.

Acknowledgment

Special thanks to Mei-Hui Su for reorganizing the tutorial notes into this paper.

References

[AHO 72] Aho, Alfred V. and Jeffrey D. Ullman, 1972, *The Theory of Parsing, Translation, and Compiling*, Vol. 1: parsing, Prentice-Hall, Englewood cliffs, N.J.

[AHO 86] Aho, Alfred V., Ravi Sethi and Jeffrey D. Ullman, 1986, *Compilers : Principles, Techniques, and Tools*, Addison-Wesley, Reading, Mass.

[ALLE 87] Allen, James, 1987, *Natural Language Processing*, Benjamin/Cummings Publishing, Menlo Park, Ca

[EAST 81] Eastman, C.M. and D.S. McLean, 1981, "On the Need for Parsing Ill-formed Input," *American Journal of Computational Linguistics*, Vol. 7, No. 4, PP. 257.

[GODB 82] Godby, C.J., R. Wallace and C. Jolley, 1982, *Language Files*, Dept. of Linguistics, The Ohio State University, Ohio.

[GORS 86] Gorsz, B.J., K.S. Jones and B.L. Webber, ed., 1986, *Readings in Natural Language Processing*, Morgan Kaufmann Publishers, Los Altos, Calf.

[HIRS 87] Hirst, Graeme, 1987, *Semantic Interpretation and the resolution of ambiguity*, Universtiy Press, Cambridge, Great Britain.

[HUTC 86] Hutchins, W.J., 1986, *Machine Translation: Past, Present, Future*, Ellis Horwood Limited, West Sussex, England.

[MARC 80] Marcus, Mitchell P., 1980, *A Theory of Syntactic Recognition for Natural Language*, MIT Press, Cambridge, Mass.

[NAGA 88] Nagao, Makoto, Panel Organizer, 1988, "Language Engineering: The Real Bottle Neck of Natural Language Processing," *Proceedings of the 12th international conference on Computational Linguistics*, Vol. 2, PP. 448–453.

[NIRE 87] Nirenburg, Sergei, ed., 1987, *Machine Translation*, Cambridge University Press, Cambridge, Great Britain.

[SU 87] Su, K.Y., J.S. Chang and H.H. Hua, 1987, "A powerful Language Processing System for English-Chinese Machine Translation," *Int. Conf. of Chinese and Oriental Language Computing*, Chicago, ILL. PP. 260–264.

[TOMI 87] Tomita, Masaru, "An Efficient Augmented-Context-Free Parsing Algorithm", *Computational Linguistics*, Vol. 13, No. 1-2, Jan.-Jun. 1987. PP. 31–46.

[WINO 83] Winograd, Terry, 1983, *Language as a Cognitive Process.* Addison-Wesley, Reading, Mass.

[WINS 84] Winston, Patrick Henry, 1984, *Artificial Intelligence.* Addison-Wesley, Reading, Mass.

[WEIS 83] Weischedel, R.M., N.K. Sondheimer, 1983, "Meta-rule as a Basis for Processing Ill-Formed Input," *American Journal of Computational Linguistics*, Vol. 9, No.3-4, July-December 1983. PP. 161–177.

[YONE 88] Yonezawa, Akinori and Ichiro Ohsawa, 1988, "Object-Oriented Parallel Parsing for Context-Free Grammar," *Proceedings of the 12th international conference on Computational Linguistics*, Vol. 2, PP. 773–778.