Performance and Usability in a Natural Language System

David R. L. Davies

Abstract

This paper describes a Natural Language Agent System that has been designed with the goal of maximising usability, adaptability, connectivity and performance. It incorporates a range of novel and state-of-the-art Software Engineering techniques to help overcome the innate complexities of natural language processing. It also provides runtime learning of new grammar and domain rules and access to local operating system and external Web data.

1 Introduction

Natural language processing presents the software engineer with layers of complexity. Tokenization of text - moving from a character level to a word level - is more complex that encountered in computer languages. Grammars are not only more complex and ultimately unspecifiable but need to be extensible at runtime.

The inference process needs to operate over a large rule-set or knowledgebase (KB) with minimal performance degradation as size increases. The system described here attempts to overcome these difficulties through the use of novel data structures and flexible algorithms. The parser and inference engine are discussed in Section 2.

Central to the motivation of this project is the idea that NLP can act as a prosthesis for human thought (following the medical rather than the linguistic meaning of the word) in much the same manner as computers have assisted us in mathematics. To this end we look at extending NLP in two directions: back towards the user and outward towards the world. For the former, in Sections 3.1 to 3.3 focus on usability of the graphical user interface and consider foundations for the future use of Automated Speech Recognition. Looking out, we consider connectivity with web and found data in Section 3.4.

2 System Performance

Full details of the Software Engineering features of the parser and inference engine are described elsewhere (Davies, 2006). Here we review the salient features that relate to the overall functionality of the system and its usability criteria.

2.1 The Parser

The parser can be described generally as a bottom-up, context free, cascading chunk parser (Abney, 1991, 1995). The algorithm used is illustrated in Figure 1. None of the grammar rules are hard-coded into the system. They are read in at runtime and added to a content-addressable memory that uses a pattern, or partial pattern, from the sentence being parsed as a key to access a replacement pattern.

In addition to English grammar the parser handles simple algebraic expressions, URLs and XML paths. The purpose of the latter two are discussed in Section 2.3. Parser flexibility will also allow its use in verifying tense and number agreement across a statement.

The parser produces two output structures. The first, a conventional parse tree, allows directed top-down search (e.g. '*find the noun phrase in the condition of a conditional statement*'). The second structure, a bit-coded integer representation, provides for rapid bottom-up determination of the syntactic context of individual words.

2.2 The Inference Engine

The inference engine has two operational phases illustrated in Figure 2. Prior to these an input query is transformed into a statement to be verified: e.g. '*Is Tom a cat?*' is transformed to the query statement '*Tom is a cat.*'.

In the first inference phase the KB is accessed to find either direct confirmation of a query statement, class membership (i.e. *A is a B*) or equivalence statements (i.e. *A means B*) for words or phrases in the query. Syntactic context

of the target phrase is verified using the integer parse representation.

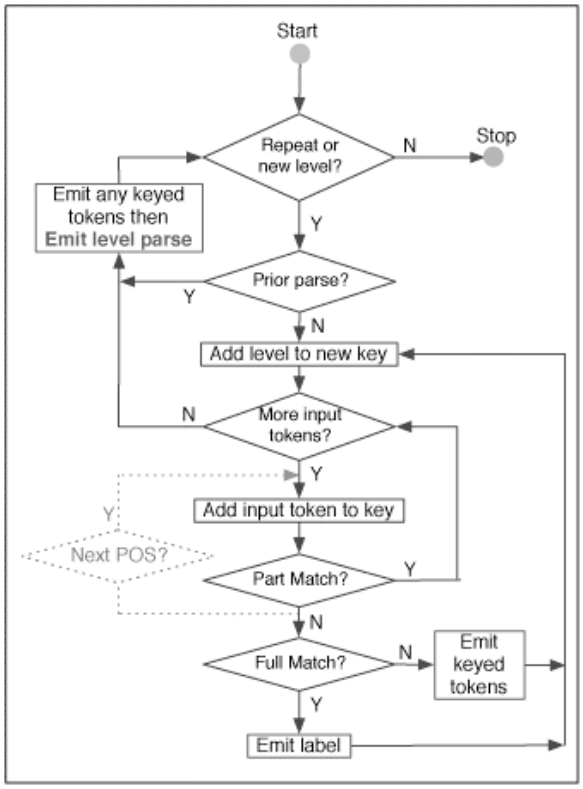

Figure 1. The Parser Algorithm

In this search phase, an inverted index structure, the Link Matrix (Davies, 1983), is used to provide optimal access speed. Only statements containing the target phrase are accessed from the KB so performance is inversely proportional to the number of relevant statements in the KB that need to be tested rather than the full KB size.

In the second phase, all alternates found in the first phase are combined in turn to create new forms of the query statement and the KB is accessed efficiently via table lookup to check for confirmation of each new statement form.

Both phases are repeated until no further progress is achieved. The process may be terminated on finding one solution but in general multiple solutions are sought to allow for the later inclusion of rule uncertainty (fuzzy logic) and consequent selection of the most probable solution.

As with grammar rules, KB rules may be added at runtime.

2.3 Context Representation

Lexical ambiguity is a fundamental problem in NLP. A low level, partial solution to the problem is provided by a mutable word class containing an iterator that allows the various meaning and part-of-speech variants of a word to be accessed at any stage in the systems processes. A similar structure is used at the sentence level.

These provide access but do not provide a means of selection. Beyond this, what is required is a means of determining the context in which the word is being used and access to ontological information specific to the current domain.

The problem of representing domain specific semantics or ontologies has been partly resolved by the inclusion of public domain XML parsers. As with the grammar parser we need both top-down and bottom-up access: top-down to allow collection of statements in a specified context and bottom-up to determine the context of a particular statement.

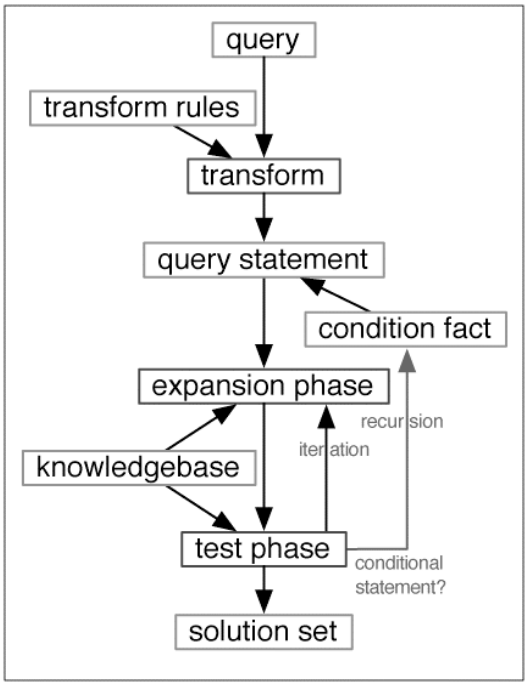

Figure 2. The inference algorithm

Since XML tags may contain attribute assignments (e.g. '*[property=value]*') the XML paths are run through the grammar parser. They are also placed in the KB so that they can be accessed through the KB retrieval functions.

2.4 Plug-ins

Natural language encompasses both statements of fact and commands. In Software Engineering terms we are combining both declarative and procedural programming. On the procedural side a further subdivision is useful: instructions directed at the user and actions to be performed by the system. Instructions (e.g. '*Give the book to Mary*') can simply be displayed to the user.

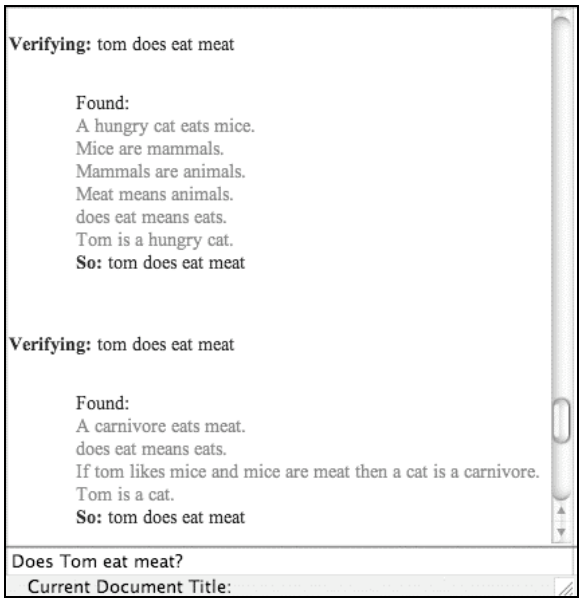

Figure 3. Display of inference results

System actions, however, (e.g. '*Display the result in window panel B*') require that the system have an expandable repertoire of Action classes. This functionality is provided via plug-ins. These are Java classes that implement a standard interface and that can be added or removed at runtime.

In addition to the Action plug-ins we have implemented a Component interface along similar lines. At this stage it is limited to GUI components. As a consequence the 'meaning' of both verbs and nouns can be extended.

3 Connectivity

3.1 The User Interface

An important advantage of using Natural Language rules, and maintaining the original sentence structure in the internal representation, is transparency. All internal processes can be displayed for the user in a relatively intelligible form. Relative that is to the Lisp, Prolog or Horn clause forms commonly used in mechanical inferencing systems.

To achieve this, however, still places a considerable burden on the design of the user interface. An initial attempt at displaying an inference trace is illustrated in Figure 3. Note that at this stage we are looking at verbal logic and the semantics of statements is ignored.

A trial parse structure display is illustrated in Figure 4. This will assist users in adapting rules to the existing grammar, developing extensions of the grammar and facilitate testing of new grammar rules.

3.2 Automated Speech Recognition

NLP and ASR are mutually supportive technologies. NLP needs ASR to broaden its domains of application (e.g. telephone access). ASR needs NLP to provide syntactic and semantic support for recognition strategies.

Davies (2002) describes an approach to ASR that also requires an intelligent controller to adapt signal processing and representational strategies. The performance demands in this operational context are high but initial tests indicate that both the parser and inference engine are capable of achieving this goal.

At its first presentation the time to parse the sentence '*If Tom is a carnivore then he eats meat.*' on an 1800Mhz PowerPC is 1.5ms however much higher performance is generally achieved since once a particular pattern has been parsed subsequent presentation is resolved by a simple table lookup.

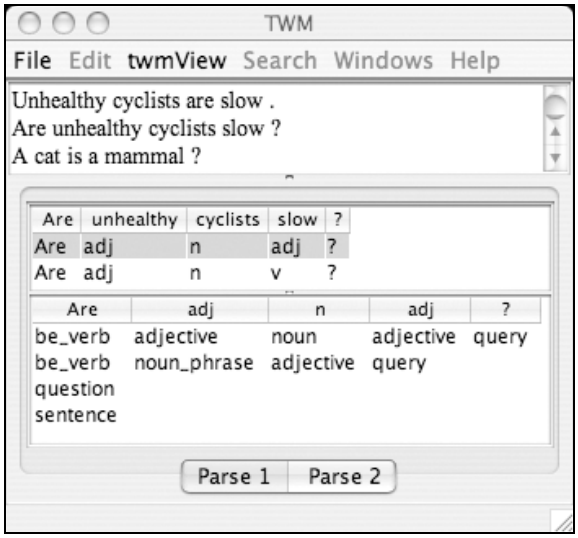

Figure 4. Parse display

Performance requirements for the inference engine are more demanding. Tests on a 400Mhz PawerPC showed speeds of approximately 6ms per inference step or around 60% of the processing time available for real-time analysis of a 10ms speech frame. Use of higher performance computers and finite state machines with direct table lookup for common signal classes may be enough to provide useable performance.

3.3 Web Connectivity

User directed extension of the KB and the potential for semantic search capability is provided via web browser functionality illustrated in Figure 5 which shows a prototype developed by student IT project groups. Two modes of operation are

provided: access to plain text files in networked file systems and access to html documents across the web.

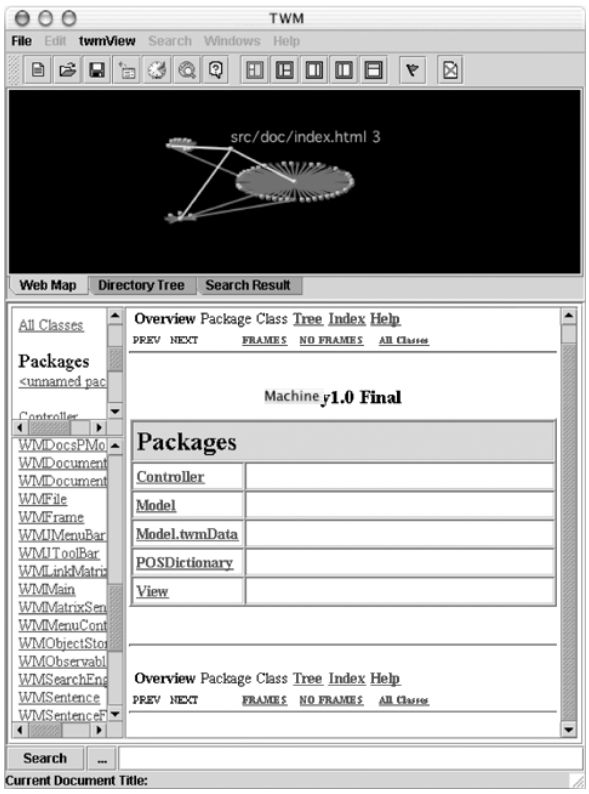

Figure 5. The web interface

To avoid needless duplication of large KBs client-server operation is required. This has been implemented, again as a student project, using public domain `xlmrpc` technology. Any instance of the system can act as either client or server.

It is envisaged that plug-ins might also be exchanged in this manner but the security issues associated with passing plug-ins across the internet have yet to be resolved.

4 Conclusion

Although incomplete, the system described here has reached a stage that might be considered 'proof of concept'. Much work still needs to be done in refining and extending the grammar, full implementation of morphological functions for tense and number and the development of meaningful rule sets to replace the test sets.

While no systematic performance evaluation of the system has been conducted at this stage it is worth noting that the inference performance for just 13 rules, reaching 3 distinct results in less than 10ms on a low performance machine, compares well with human performance on the same task of around 100 seconds – a difference of four orders of magnitude. Since system performance drops approximately linearly with KB size the performance difference can be expected to increase dramatically as human performance drops rapidly with large KBs.

Acknowledgement

The author gratefully acknowledges the valuable contributions made by many colleagues, friends and students in achieving the results obtained so far. Particular appreciation is due to Author Zawadski for his work on the prototype, Xiaoming Zhang for his contribution to the browser package and the design and construction of the trial parser display and to Darren Hitchman. for the client-server package.

References

Dave Davies. 1983. *A Data Structure for Text Compression*, Australian Microcomputer Software Conference, ACS, Canberra.

Dave Davies. 2002. *Representing Time in Automated Speech Recognition*. PhD thesis, www.blis.canberra.edu.au/hccl/StaffPages/DaveD/DaveD-Thesis.pdf.

Dave Davies. 2006, in preparation.

Steven Abney. 1991, *Parsing by Chunks*, in Robert Berwick, Stephen Abney, Carol Tenny (eds.), *Principle-Based Parsing*, Klewer Academic Publishers.

Steven Abney. 1995, *Partial Parsing via Finite-State Cascades*, Natural Language Engineering 1 (1), Cambridge University Press.

COLING/ACL Interactive Demonstration Script

Aims of the Presentation

The overall aim of the presentation is to generate interest in a system that, it is hoped, will provide a useful test bed for the practical application of techniques in computational linguistics for researchers and a teaching tool for students.

Specific aims include demonstration of::

- The practical value of using Natural Language as the foundation for rule-based agent systems that comes through transparency of the internal operations.
- The practicality of combining declarative and procedural language functions.
- The current development status of the system
- The advantages of using highly interactive techniques to overcome some of the difficulties of natural language processing
- Use of state-of-the art development environments in NLP
- Use of novel, highly efficient data structures and software engineering techniques in NLP
- The value of system, network and Web connectivity in extending the practical application domains of agent systems.
- The value of adaptability through user runtime extension of both the system grammar and the knowledgebase.

The Presentation:

Introduction: general aims of the project and this presentation of the system, its limitations and constraints

A brief history of the project and acknowledgement of its principal participants

Complexity: layers of complexity and how to deal with it.

The parser operation.

Diagram of the parser algorithm
Live demonstration of the basic functionality
Overview of the grammar and invite comments on possible limitations, modifications and extensions.
Invitation to audience to suggest sentences to test the parser – given the limitations of the current grammar.
Demonstration and discussion of the prototype parse display (separate package)
Performance statistics

The inference engine operation

Diagram of the inference algorithm
Live demonstration of the basic functionality
An invitation to the audience to suggest new rules and queries to be entered and tested
Performance statistics

A discussion of the user interface of the parser and inference engine.

Invite suggestions and feedback

The web access package (student project)

Live demonstration of functionality (using local web pages)
Screenshots of an alternative approach

A brief overview of the client-server package (student project)

Discussion of problems encountered and their solutions - where solved.

Future directions, intended initial application – a project management toolkit that will provide the system with general problem solving and task management capability.

Discussion

End of presentation

Presentation Requirements:
Power and an overhead projector or comparable large screen display.