UNIVERSITY OF DURHAM: DESCRIPTION OF THE LOLITA SYSTEM AS USED IN MUC-7

Roberto Garigliano, Agnieszka Urbanowicz and David J. Nettleton.

Laboratory for Natural Language Engineering,
Department of Computer Science,
University of Durham,
South Road,
Durham. DH1 3LE
United Kingdom
email: R.Garigliano@durham.ac.uk

INTRODUCTION

This document describes the LOLITA system and how it was used to run the MUC tasks of named entity, co-reference and template element. Details of the system's performance are given for the walk-through articles as well as overall performance.

LOLITA has been designed in such a way that the code implementing the MUC tasks is only a small part of the whole system. A core system provides complex facilities with the MUC system being built so that it utilises these facilities. Hence, after some background to the LOLITA project, the 'core' of LOLITA is described. This system description is substantially similar to that given for MUC-6 [1]. However, some important changes to the underlying core system have occurred since MUC-6 and these are presented in their own section. Following this is a description of changes that were required *specifically* for MUC-7.

An analysis of the system's performance for the walk-through articles is presented together with an overall view of the system's performance. Prior to the conclusions a section on further and ongoing work is provided. This includes brief descriptions of some important work that is being undertaken, in particular: a re-engineering of the system which will result in a C++ version, and the addition of a large number of dictionary word definitions to the knowledge base.

BACKGROUND

The LOLITA (Large-scale, Object-based, Linguistic Interactor, Translator, and Analyser) system is designed as a general purpose Natural Language Processing (NLP) system and has been under development at the University of Durham since 1986. The system is designed to provide NLP capabilities to support many applications in multiple domains. It attempts to do this by providing a core platform upon which different applications can be built. This core platform provides two main facilities: analysis, which converts text to a logical representation of its meaning, and generation, which expresses information represented in this logical form as text. Unlike many of its contemporary NLP systems, the LOLITA system is not designed as a framework that can be tailored to specific domains, but as a system that brings its knowledge of specific domains to bear as and when appropriate.

The Laboratory for Natural Language Engineering (LNLE) at the University of Durham is focussed on developing this core. Prototype applications have been built using the core facilities; some of them are listed below:

- Information extraction: production of summary and other templates.
- Simple meaning-based translation: currently Italian to English.
- Natural language query: supplying information to LOLITA and then asking questions about this information.

- Dialogue: a model of dialogue has been implemented.

- Chinese language tutoring: a mixed English and Chinese grammar allows detection of students of Chinese using English constructions, and diagnosis of problems.

The MUC competitions have provided an opportunity for the Laboratory for Natural Language Engineering to evaluate the approach used in the LOLITA system on some very specific tasks as well as a chance to strengthen the system's performance in the domain of newspaper articles. Given the wider aims of the project, the approach taken was to put minimal effort into the development of the new applications needed for the MUC tasks and maximum effort into the development and improvement of the core system.

The following three sections describe the details of the LOLITA core as used for MUC-7. The description of the architecture in the following section is substantially similar to that for MUC-6 and readers familiar with this may choose to skip over most of this section. The section following 'Architecture' describes some important modifications that have been carried out on the core since MUC-6. The majority of the time spent in preparing for MUC-7 was spent enhancing these pieces of core functionality, and developing appropriate rules and knowledge bases for the tasks. This approach follows from the design that was adopted, *i.e.*, don't tailor the system to the particular domain, instead develop core functionality and use the available rules/knowledge where appropriate.

ARCHITECTURE

Overview

LOLITA is designed as a core system supplemented with a set of applications, the former supplying basic NL facilities to the latter. Figure 1 shows the MUC-relevant parts. The most important part of the core is the large knowledge base, which is called the 'Semantic Network', SemNet or net, for short. It is heavily used in most stages of analysis, and the results of analysis are added to it, as a disambiguated logical representation of the input. The analysis stages are fairly standard, and are arranged in a pipeline. Each is implemented in a rule-based way. The system does not currently use any form of stochastic or adaptive techniques in the main system.

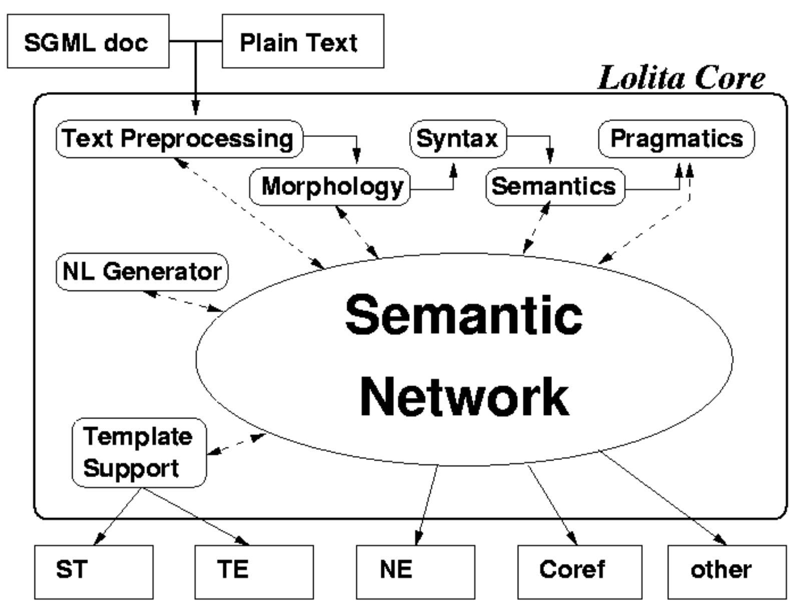

Figure 1: Block diagram of the LOLITA core plus some applications

The applications can then read the results of the analysis from the SemNet, and generally interrogate the contents of the SemNet. Some central ‘support’ facilities are provided to aid application writing, such as the general template mechanism and the NL generator - which translates pieces of the SemNet into English. More detail on the architecture of LOLITA can be found elsewhere [2].

The Semantic Network

The SemNet is a 100,000+ node, directed hyper-graph. Each node has a set of links, plus a set of ‘control variables’ (or *controls*). Some nodes have an associated ‘name’: this is usually a single word which characterises the meaning of the node. Each link has an arc and a set of targets. Targets are other nodes, and the arc too is just a node. Nodes correspond to concepts of entities or events. Links correspond to relationships between nodes. Since an arc is also a node, the concepts of the different kinds of relationship possible between nodes can be represented in the same formalism as more concrete concepts. In this system, the ‘meaning’ of any particular node is given by its connections, its relative position in the net.

Controls indicate basic information about a node, such as its type (*e.g.*, event, entity, relation), its family (*e.g.*, human, inanimate, food, organisation), its lexical type (*e.g.*, noun, preposition, adverb) - as appropriate. An important control is a node’s *rank*: this encodes quantification information. Concepts of general sets have a *Universal* rank, specifically named objects have a *Named Individual* rank, and general individuals an *Individual* rank. There are several other less important ranks, used for things like encoding script-like information or existential quantification. Controls could be represented using links, but for efficiency reasons a more compact form is used.

There are approximately 60 different arcs. The arcs subj_, action_, and object_ are used to represent the basic roles of an event. Events can have other arcs, such as those indicating temporal information, the status of the information (*e.g.*, known fact, hypothesis, etc), or arcs that indicate the source of the information. Most arcs also have inverses: *e.g.*, the subject_ arc has the inverse subject_of_, which allows determination of the events in which a particular concept played the subject role.

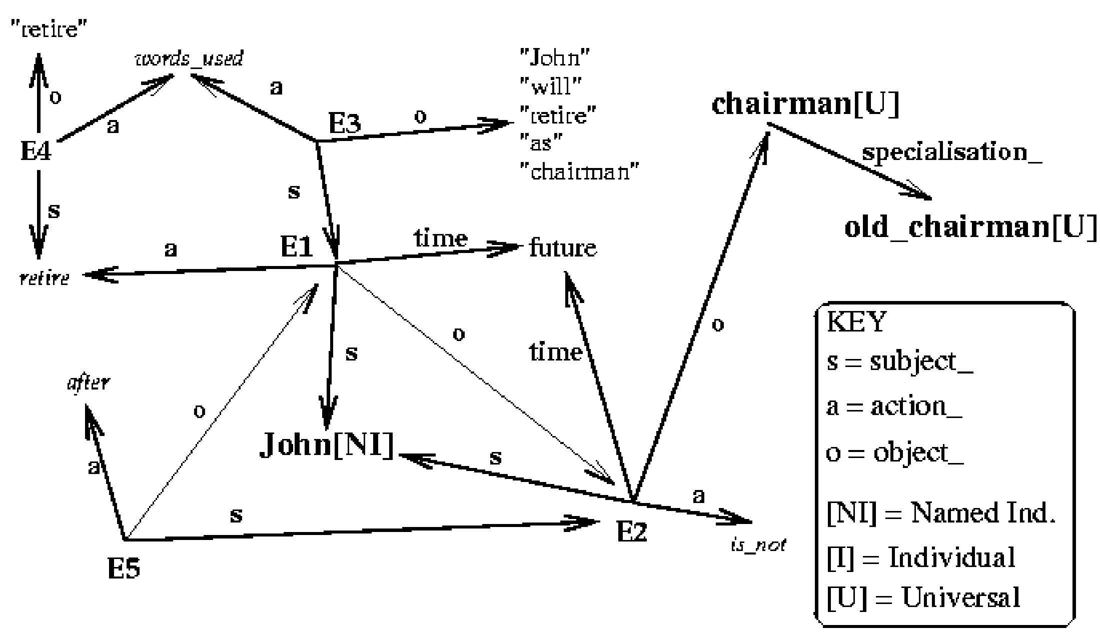

Figure 2: Example piece of semantic net, for the sentence "John will retire as chairman". It is given here as an example of SemNet structure, and its meaning is discussed in the section on the semantic network. The full structure is not shown, for reasons of space.

Concepts are connected with arcs such as specialisation_ (and its inverse, generalisation_), or instance_ (inverse universal_). Specialisation links a set to one possible subset; for example, in Figure 2, chairman[U] represents the set of all possible chairmen, and old_chairman[U] the set of all possible old chairmen. Between the former and the latter is a specialisation_ link, indicating that old chairmen are a subset of chairmen. Conversely, the latter is linked to the former with a generalisation_ link, representing a superset. Using the specialisation_ link, hierarchies of concepts are specified. The instance_ arc connects a concept to an instance of that concept: *e.g.*, a particular chairman chairman1[I] would be linked to chairman[U] by an instance_ link. Other links between concepts include synonym_ and antonym_.

The SemNet is used to hold several kinds of information:

- Concept hierarchies: built with arcs such as generalisation_, concept hierarchies encode knowledge like "man is_a mammal is_a vertebrate" etc. They prevent duplication of information by allowing information to be inherited within the hierarchy.

- Lexical information: actual words are represented in the net, and their properties are stored in the net, as opposed to having a separate lexicon. The lexical-level nodes are indexed via a simple dictionary: *i.e.*, a mapping from root words to all the senses of that word. Note that the lexical forms are distinct from the concepts: they are linked by a concept_ arc. Concepts are linked to lexical forms by a link named after the language of interest. For example, dog[U] has a link english to the noun form of 'dog', and a link italian to the Italian word 'cane'.

- Prototypical events: these define restrictions on events by providing 'templates' for events, *e.g.*, by imposing selectional restrictions on the roles in an event. "Human owners own things" says that only humans can take the subject role in 'ownership' events.

- General events: other kinds of information. For example, the content of a MUC article would come in this class, when analysed.

The bulk of the net (70%) comes from WordNet, a database containing lexical and semantic information about word forms in English [3]. More details about the formalism used in the net can be found in [4].

Referring back to the Original Text

Before MUC-6, LOLITA did not have a method of referring back to its input: the previous orientation was to move from language-dependent surface forms to a language-independent logical representation. Therefore, information about the surface form was discarded. Since the ability of reference has many uses outside of the MUC tasks, a more general mechanism was designed and added to the core. It allows fine-grain connection of the analysis results to the sections of the document giving rise to those results. The system allocates new SemNet nodes to components of the document (words, phrases, sentences, ...), which act as references into the document. This is called the 'Textref' system.

Textrefs allow the document structure to be fully represented in the net, and represented uniformly with the other information in the system. At the word level, a Textref signifies a specific occurrence of a word at a certain position in the input, and is distinct from the nodes representing the lexical or semantic forms of its root form. It is an instance_ of the universal concept of all occurrences of that word. Concept nodes and Textref nodes are linked by an event with the internal action words_used. Two examples may be seen in Figure 2: single words are attached to the 'key' words of the sentence (only 'retire' is shown), and all of the Textrefs in the sentence are attached to the node representing the whole event.

Text Pre-processing

Core analysis of textual input starts from a LOLITA-specific SGML representation of the input (called an SGML tree). Individual applications must convert from their own formats (*e.g.*, plain text, MUC articles, LaTeX, HTML, ...) into this internal format. The MUC converter is just a simple SGML parser. The preprocessor then adds additional structure to the internal SGML tree where necessary. In particular the following structures are handled in the order given: reported speech, paragraphs, sentences and words. Markers for reported speech are distributed over all sentences inside the quotes. Lastly, each word is allocated a Textref.

Morphology

Morphology is applied to an SGML tree whose leaves are individual word tokens, and whose nodes represent the structure of the document. A few transformations are done on this structure to unpack contractions (*e.g.*, "I'll" expanded to "I will"), expand monetary and numeric expressions (*e.g.*, "$10 million" to "10 million dollars"), and to transform certain surface-level idiomatic phrases (*e.g.*, "in charge of"). Some splitting of hyphenated words is also done. Then, the basic morphology function is mapped on to all leaves (with additional treatment provided for sentence initial words).

Lookups in the dictionary are done with the root forms suggested by affix stripping. If successful, a word is linked to lexical and semantic nodes, allowing access to lexical and semantic information during the rest of morphology, parsing, and semantics. Affix stripping loses information such as number and case, so this information is represented using a Feature system. Features are used in parsing (described below). Other Features include word class (Noun, Verb, ...) and some semantic-based Features. Finally, possible syntactic categories for a word are determined from the lexical (and sometimes semantic) node information. Thus, each leaf is mapped to a set of alternatives, varying in category and Features, which represent all possible interpretations of that leaf.

Parsing

The parsing mechanism utilised in MUC-6 consisted of five stages:

1. A pre-parser which identifies and provides structure for monetary expressions.

2. Parsing of whole sentences using the Tomita algorithm [5]. The result of this stage is a "parse forest", a directed acyclic graph which indicates all possible parses. Due to the complexity of the grammar, this forest is frequently very large, implying many possible parses.

3. Decoding of the parse forest. The forest is selectively explored from the topmost node, using heuristics such as Feature consistency and hand-assigned likelihoods of certain grammatical constructions. Feature errors and unlikely pieces of grammar involve a cost: the aim of the search is to extract the set of lowest-cost trees.

4. Selection of best parse tree: subsequent analysis operates on a single tree. The lowest cost set is ordered on the basis of several heuristics on the form of the tree. For example, preferring a deeper tree.

5. Normalisation: syntax-based, meaning-preserving transformations are applied to the trees to reduce the number of cases required in semantics. A prime example of this is passive to active, *i.e.*, "I was bitten by a dog" changed to "A dog bit me". Another class involves transformations such as "You are surprised" to "*SOMETHING* surprised you", which makes explicit the object doing the surprising.

Although this mechanism remains at the core of the parsing for MUC-7 a number of additional strategies have since been included. These are described below in the section detailing the main changes to the system since MUC-6.

An example parse is given in Figure 3. Note that ‘will’ and ‘as’ are missing. As so-called function words, they don’t carry much inherent semantic meaning, so the tense information of ‘will’ is transferred to the Features of the main verb, and the copula function of ‘as’ is transformed into a syntactic construct. This simplifies the semantic rules.

```
Sen                              -- sentence branch
  full_propernoun                -- proper noun phrase
    propernoun JOHN [Sexed]
  neg_copula                     -- copula verb phrase
    asCopulaN RETIRE [Fut]
    comnoun CHAIRMAN [Sing,Per3]
```

Figure 3: Parse tree for "John will retire as chairman".

Analysis of Meaning

This section describes how the parse tree is converted to a disambiguated piece of SemNet. There are two stages to this ‘semantic’ and ‘pragmatic’. The semantic analysis is generally compositional in nature: the meaning of a tree is built from the meanings of its subtrees. A mechanism goes through the parse tree in depth-first, post-order traversal, applying semantic rules mainly on the basis of the syntactic phrase type of the current tree node. If the meaning of a particular subtree is unambiguous in role, the Textrefs for the text in that subtree are connected to that meaning. Since the meanings can be nodes which already have Textrefs connected, then particular nodes can collect Textrefs for all occurrences of their mention. This Textref handling is completely invisible to the semantic rules.

A state value, the ‘context’, is passed around during traversal: this holds possible referents in order of occurrence, and is used to resolve anaphoric expressions. Use of this context prevents the semantics being purely compositional.

The ‘meaning’ of most leaves is the semantic node associated with the word at the morphology stage. The node is passed to the leaf’s parent in the form of a ‘role’ structure, which indicates the role the node may play in the semantics of the parent. Often this is unknown, but in cases like verbs, it can be determined as the *act*. The actual role structure allows for representation of semantic ambiguity.

The main task of the pragmatic stage is disambiguation and type checking. Lexical ambiguities and anaphora are resolved using a series of preference heuristics which are first applied to disambiguate the action of the event. Once the action is known, any knowledge available from the prototype event associated with that action can be used to rule out pragmatically implausible readings, as well as to aid disambiguation of the remaining elements of the event (in the spirit of [6]).

The contents of the current context together with the topic of the text (the latter is given to the system in advance) influence the choice of word senses: those meanings are preferred which are semantically closer to the meanings present in the context or the topic, where semantic closeness is computed on the basis of the distance between nodes in the network. Other factors may cause one concept to be preferred over others, such as the amount of knowledge the system has about a given concept, or the concept’s frequency of use.

Once an event is disambiguated, the system attempts to establish plausible connections between it and the previously processed discourse.

Reference Resolution

As the discourse is processed, the referents found in it are stored in the 'Context' buffer. Each time an anaphoric expression is identified in the incoming discourse, the system looks for a possible referent for this expression in the Context buffer (obeying matching rules dictated by the type of anaphoric expression). If the system finds no match, it introduces a new entity into the Context. If the system finds just one match, it unifies the two and adds the newly unified item into the Context. If the system finds more than one match, it builds a special structure to represent the ambiguity and passes it onto the system of preference heuristics to decide between the possibilities.

The heuristics are loosely based on ideas from centering theory [7], psycholinguistic findings as well as common sense. They assess the salience of the candidates based on grammatical and semantic features, as well their position in the sentence, recency of mention and relatedness to the topic of the text. As in the whole of the LOLITA system, the algorithm relies heavily on a correct parsing and semantic analysis.

Template Support

The processes involved in producing templates can be generalised, hence the core contains a mechanism to help write templates at an abstract level. This mechanism handles search through the net, use of inference rules to derive implicit facts, and general output formatting.

A template contains a predefined set of slots with associated fill-in rules that direct the search for appropriate information in the net. The slot fill-in rules are predicates that check node controls, or use the inference functions available in the core. For more details see [1].

Implementation and Operating Details

LOLITA is written mostly in Haskell, a non-strict functional programming language [8]. Two resource-critical sections are written in C - the parsing algorithm and the SemNet data structure and its access functions. Haskell has some similarity to LISP, such as building programs by writing functions, a garbage-collected heap, lists as a basic type, and full higher-order use of functions. However, it provides excellent support for modern Software Engineering, such as modularity, constrained polymorphism, a strong but flexible type system. It also enforces referential transparency and allows coding in a 'lazy' style, which means code is not executed unless needed. Thus, whilst our system has the external appearance of a pipeline architecture, the evaluation of individual pieces of code need not occur in that strict order.

IMPROVEMENTS CARRIED OUT SINCE MUC-6

The system that entered the 6th Message Understanding Conference suffered from three major problems. First, there was room for improvement in parsing. Second, the named entity recognition rate was fairly low, as compared with other systems. Third, the system contained a series of trivial errors in the code. Altogether, these three major shortcomings resulted in a considerable drop in performance.

In the general approach adopted by the LOLITA project, every core component plays an important role in the final result. Consequently, if any of the components is unsatisfactory, overall performance is affected. This is especially prominent if an early stage of analysis (*e.g.*, parsing) is incorrect.

Many of the problems encountered during MUC-6 have been addressed and several improvements to the system have been carried out since that time. The most substantial of these improvements are discussed in the remainder of this section.

Changes to the Grammar and Parsing Components

Parsing can sometimes fail on very large forests: decoding these requires a lot of resources (time, memory). Rather than cause a crash due to overrunning limits, the parse is abandoned. This is implemented by fixing a time-limit on the process - resource usage being proportional to time: the expiry of the time limit is referred to as a 'timeout'. It is also possible for parses to fail if the sentence can't be analysed with the main grammar. In the system used for MUC-6 if the parse failed then analysis was discontinued on that sentence. This meant that no semantic result was produced and hence no information was available on NE's etc in the sentence. MUC-6 texts which contained sentences that timed out would therefore receive poor scores. For MUC-7 a number of improvements to the parsing mechanism have been adopted, including a recovery strategy for sentences that failed to parse.

LOLITA's grammar has been improved and expanded to allow for a better parsing of the materials used in MUC-6. Furthermore, a new method for handling headlines in the articles has been added. As well as devising a special grammar for them, the headlines are now analysed at the end of the article, using as context the initial sentences of the main body of the text.

The parsing mechanism itself has been improved. Island parsing, whereby easily recognisable noun phrases are 'locked' into units before being passed onto the parser, has been introduced. This has improved the parsing success rate substantially.

Moreover two extra passes have been added should parsing fail: a second pass using Brill's tagger [9] and a third pass using a reduced grammar. These are aimed at recovering constituents of complex sentences, if a full parse isn't possible.

Finally, in cases where all three parsing passes fail, a way of recovering most named entities, all the pronouns, possessive determiners and some noun phrases (particularly those related to the topic of the text, if the latter can be provided in advance) has been devised.

Changes to the NE Recognition Components

The components responsible for named entity recognition have been revised and many new rules have been added. A major change has been introduced to LOLITA's morphology module, which allows the system to reuse names of entities previously recognised in the preceding text, rather than treat the entities in each sentence of the incoming text as new. (Previously, the morphology module had no access to the results of the semantic and pragmatic analysis of the preceding text.)

A change in the treatment of unknown proper names that appear without clear designators (*i.e.*, without *Corp*, *Ltd*, *Mrs.*, *etc*) has been introduced. In the MUC-6 system a decision as to what type of entity an unknown name stood for was made early and usually resulted in the conclusion that it must stand for an organisation. The new improved treatment, on the other hand, involves the introduction of the concept of human_or_organisation, the use of which allows for a delay in the decision, until some disambiguating information becomes available at the pragmatics stage. For example, given the following first sentence of an article:

Shortly after Fossett's launching Monday his competitors sent him telegrams of congratulation

The system cannot decide what sort of entity *Fosset* is on the basis of this name itself. However, the use of the pronoun *his* as well as the absence of any other possible referents, provide the disambiguating clues.

Changes to the Semantic and Pragmatic Components

At the semantic level, several new rules that had previously been missing and had been needed to handle expressions common in MUC-6 articles have been added. New rules were also needed due to the introduction of new constructions in the grammar.

In the pragmatics component, the preference heuristics system has been substantially revised and expanded. In the MUC-6 system the heuristics acted as filters and so rejected any non-preferred candidates. This sometimes resulted in rejecting a candidate which didn't match one of the heuristics that was applied at an early stage. The same candidate could have been favoured by several later heuristics, but this had been ignored.

Currently, the preference heuristics assign penalty points to non-preferred items and at the end of their application, the candidate with least penalties is chosen as the referent.

Increase in Basic Data

Since the time of MUC-6, a lot of data concerning organisation names, corporate designators, personal names and place names have been added to LOLITA's knowledge base (SemNet). The additions include names of major USA institutions and organisations (*e.g.*, government departments), names of newspapers, names of major geographical locations in the USA, US states abbreviations and names of countries and nationalities of the world. Also, about 8000 new forenames have been added and all the existing forenames have been checked to ensure that they are marked correctly for gender.

Text Output Errors Corrected

Minor coding errors in the 'Textref' module of LOLITA resulted in the system occasionally inserting spurious space characters in some places, while deleting others. This adversely affected the final result of the MUC-6 evaluation, because the scoring software is sensitive to any misalignments between the answer keys and the responses. Many of these kinds of errors were corrected before participation in MUC-7.

MUC-7 SPECIFIC CHANGES

The work carried out during the preparations for MUC-7 was concentrated in four main areas. These are discussed in this section.

Addition of MUC-7 Specific Data

Over three hundred airline names as well as some well known airport names have been added to the SemNet. Additionally, airline and aircraft specific artifacts, such as types of aircraft and most common aircraft models (including some military ones), have been added. The area of the SemNet with knowledge relevant to the aircrash scenario has been checked and adjusted as necessary.

Grammar Expansion

Grammar rules have been added to deal with constructions common in the training texts, such as references to aircraft and flights (*Boeing 747*, or *Paris-bound Boeing 747*, *the TWA flight 800*, *etc*).

The MUC-7 corpus contains sentences which are, on average, much longer than the ones encountered in most of our previous tests. Sentences of around 40 words are not uncommon. In view of this the island parsing and the third pass parsing (of fragments of sentences, using a reduced grammar) proved particularly important, hence, a reasonable amount of work was needed on the rules for the reduced grammar and the failure recovery mechanism.

Revision of Pragmatic Disambiguation Rules

In order to deal with reported speech, commonly found in the training articles, it was necessary to improve the rules and heuristics in LOLITA which deal with first person pronouns (*i.e.*, 'I' and 'we'). However, the new rules that were introduced using the examples from the MUC-7 training corpus were not designed specifically for those examples. In line with the normal strategy of the LOLITA project, our intention was to make them as general as possible.

It was also found that some of our existing rules for noun phrase matching were not working well with the MUC-7 corpus. The existing rules produced much better results for the MUC-6 corpus whose topic area generally involved only companies and people. The rules needed tightening, especially when dealing with references to locations and aircraft related artifacts.

A certain number of rules that we introduced were very MUC-7 specific and conflicted with LOLITA's basic analysis. For example, in a sentence such as:

> *The military version of the Boeing 737 that crashed in Croatia Wednesday was not equipped...*

Boeing was to be marked as ORGANIZATION, while in LOLITA's analysis, it is an artifact.

Rules to Handle 'Non-Natural' Language

Special treatment had to be devised for PREAMBLE and SLUG fields of the articles at the morphology and parsing levels. Some of these fields contained strings which seemed more like a code (particular to the New York Times News Service) than natural language, for example:

<SLUG fv=ttj-z> BC-LUCID-STUCK-HNS </SLUG>
<SLUG fv=tia-z> BC-JAPAN-ROCKET-HUGHES-B </SLUG>
<SLUG fv=tia-z> BC-CLINTON-CHINA-SATELLI </SLUG>

<PREAMBLE>
BC-WALSH-OBIT-NYT
(Fla., Mass., N.J., Md., Colo., R.I. ATTN)
JOHN PAUL WALSH, 78, FORMER SPACE SCIENTIST
(lb)
By WOLFGANG SAXON
c.1996 N.Y. Times News Service
</PREAMBLE>

Additional morphology and grammar rules had to be written specifically to handle these. The system processes them at the end of the analysis of the main body of the text in the hope that the text can provide a useful context in which to deal with them.

RESULTS

The Named Entity Task

The system's total score for the 100 articles of the formal run was:

P&R 76.43 2P&R 77.31 P&2R 75.57

This is an improvement on the scores for the named entity task that the system achieved during MUC-6. However, the score was a little disappointing, as during training the system consistently achieved scores in the region of mid to high eighties.

A shift in the topic from airlines and aircraft to satellite, rocket and missile launches explains some of the problems that were encountered. LOLITA's data in the latter area was not strong. Apart from the specific data, some basic data was found to be missing too, *e.g.*, the names of the planets of the Solar system. Furthermore, the system was not prepared to recognise space shuttle names (*e.g.*, *Endeavour*, *Columbia*) or missile names like *Scud*, *Patriot*, *etc*.

A number of company names in the satellite television market were also missing. Several of them come without a clear designator and were not recognised by the system, *e.g.*, *BSkyB*, *SatelLife*, *Intelsat*, *Comsat*, *Canal-Plus*, etc. (NB. *BSkyB* might have been resolved correctly in the presence of *British Sky Broadcasting Group Plc* in the same article. Unfortunately, the rules in the acronym matching algorithm didn't handle this case correctly.)

Finally, a mistake was made in interpreting the guidelines of the task. The names of newspapers were not marked as ORGANIZATIONS and so this too contributed to a drop in scores.

Walk-through article

The score for the walk-through article was:

P&R 75.57 2P&R 76.63 P&2R 74.55

This is very slightly lower than the overall score for the formal run. The worst scoring group of entities in this article was ENAMEX PERSON, where out of 16 entities, just over half were marked correctly (R 56%, P 53%). This is different from the overall trend, where the system's score for PERSON was a lot higher (R 80%, P 74%).

An example of an error that occurred in this text is for *Llennel Evangelista*. The sentence:

> *Llennel Evangelista, a spokesman for Intelsat, a global satellite consortium based in Washington, said the accident occurred at 2 p.m. EST Wednesday...*

was incorrectly analysed for two reasons. First of all the name *Llennel* was not in the data base. Secondly, a parsing error resulted in the analysis of *Llennel Evangelista* as both *a spokesman* and *a consortium*. The label of ORGANIZATION was then (incorrectly) chosen.

Although it is helpful for the system to have the names of people in its database, it is not crucial. For example, the name *LaRae Marsik* was also unknown to the system, but this was dealt with correctly, because here the correct parsing facilitated a correct analysis:

> *A spokeswoman for Tele-Communications, LaRae Marsik, said the partners in the Latin American venture intended to begin service by the end of 1996.*

So, because *LaRae Marsik* was understood to be a spokeswoman (a concept known to the system), it was possible to conclude that it must be a PERSON.

A number of named entities in the SLUG and PREAMBLE fields were missed, *e.g.*, three occurrences of *MURDOCH*. The strings in these fields appeared to be in some special format rather than natural language. The rules required by the system for handling these fields were therefore rather specialised. As these were MUC-7 specific, relatively little effort was spent polishing them; the effort instead being expanded on the more generally useful core rule sets.

The recognition of organisations was much better in this article: R 64%, P 80%. This is higher than the overall results in the whole test (the score there was R 63%, P 67%). The most common reasons for errors appear to be problems with tokenisation or parsing.

The Co-reference Task

The LOLITA system's score for the 20 articles in this evaluation was:

Recall 46.9% Precision 57.0% f-value 51.5%

As with the named entity task, this was a much better score than the system achieved in MUC-6.

One of the reasons leading to co-reference resolution errors was lack of full parsing. The performance of the co-reference resolution task within our system is sensitive to the basic analysis being correct: possibly more so than the other two tasks in which the system was entered. In many cases, the full parse was not available and the parsing recovery mechanism didn't always provide sufficient input to facilitate a good semantic analysis. This led to problems, whereby even seemingly easy co-reference links were sometimes lost.

Another problem was the fact that due to lack of resources not enough time was devoted to dealing with co-references involving conjoined noun phrases. In the previous MUC such co-references were excluded by the task, and although the LOLITA system has never explicitly excluded co-references involving conjoined noun phrases the rules that were used required more thorough testing than resources allowed.

Some trivial errors also contributed to a reduction in score. For example, in the document 9601160264 the string *McDonald* was consistently marked instead of *McDonald's*. Additionally, due to a text output error, a second chain containing some occurrences of *McDonald* was built. This lowered the score considerably. Having corrected the error, the score for this article increased by just over 10%. This increase leads to a slightly better overall score of:

Recall 48.0% Precision 58.6% f-value 52.8%

In a number of articles the system scored particularly well: 6 of the articles scored well above the f-value of 60% and one of them as high as 70.5%. The problems that were encountered in other articles were typically due to the lack or resources that were available for debugging and testing. The developers believe that with a modest amount of further effort the system's overall scores for this task would have been even higher.

The problem such as the one illustrated by the *McDonald* example points to a difficulty in automatic scoring and evaluation. On a semantic level, the system connected together the correct chain, however for scoring purposes this constituted a spurious chain. The resulting drop in score is treated the same as other spurious connections that could have been semantically incorrect. A scorer which is able to include a semantic component would give a more accurate reflection as to the success of any 'deeper' analysis that a system may have undertaken.

Walk-through article

The official score for the co-reference walk-through article is:

Recall 45.6% Precision 57.1% f-value 50.7%

This was slightly lower than our system's overall score for this task.

Although performing reasonably well on most of the smaller chains within the article, problems occurred with two of the longer chains. The first involving *Hughes* and the second with *Federal Communications Commission/FCC.* In the case of *Hughes*, problems with the analysis of *Hughes' Galaxy VIII(I)* led to losses in both recall and precision. The string *Hughes* was not marked up at all, while *Galaxy VIII(I)* was split into two separate units *Galaxy VIII* and *I*.

We noticed also that in one case our system marked a larger maximal noun phrase than the key. Having changed the key (according to what we believe is consistent with the task description) to include the following as an antecedent:

> ... the Federal Communications Commission's allocation of aswath of spectrum that will let their earth stations communicate with satellites in space

rather than just:

> ... the Federal Communications Commission's allocation of a swath of spectrum

we gain some extra points in the score: Recall 46.8%, Precision 58.7%, f-value 52.1%.

Template Elements Task

The overall score on this task was:

P&R 66.75 2P&R 69.74 P&2R 64.01

This result is probably the most satisfying of the three MUC tasks that the system entered. Using LOLITA's template support it was possible to produce very reasonable templates within a relatively short space of time. To prepare the system for this task took only about 10 person days.

The best performing subtask was the entity slot, particularly where the task required the extraction of organisations and persons. The lowest score was obtained in the ENT_DESCRIPTOR category, which could have been increased, had the co-reference performance score been better.

Walk-through article

The walk-through article score was:

P&R 76.92 2P&R 77.05 P&2R 76.80

which is better than the system's overall score. The errors made were mainly due to tokenisation, for example, in *International Technology Underwriters of Bethesda, Maryland* the system didn't take the strings as a full name of company but split off *Maryland* into a separate entity. There was a similar problem with *Space Transportation Association of Arlington, Virginia.*

Also, some inaccurate co-reference resolution resulted in spurious ENT_DESCRIPTORS: for example, *the company's Washington headquarters* for *Bloomberg* and *Rupert Murdoch's* for *News Corporation*. The latter descriptor error is much less serious than the former, as it does actually make some sense. The former descriptor, however, is erroneous.

Other errors are data dependent: for example, the system didn't have some basic geographical data for well known cities of Europe, resulting in *Paris* being classified as a region.

Despite the above problems the majority of the underlying analysis for this text was correct. The system was then able to successfully apply the higher level heuristics at the semantic and pragmatic levels. This demonstrates that given a correct underlying analysis the development of the high level template element application is relatively trivial. Results such as these provide further evidence to the developers that concentrating development on the core analysis is a strategy that will produce the best long term results.

CURRENT ACTIVITIES

Re-engineering of the System

The use of Haskell has proved very beneficial in the development of the LOLITA system (see [1] for more details). Of particular benefit is the ability to quickly prototype complex algorithms. More recently the need for a large amount of prototyping has diminished, as the core parts of the system have become relatively stable. In order to improve the system's performance a major effort is now underway to re-engineer large parts of the system in C++. This has also provided the developers with the opportunity to make some alterations to the structure of the overall system.

The aim of the project is still to develop a powerful core set of tools (*e.g.*, parser, *etc*) which can then be used by a number of high level applications. However, the re-engineering process has also offered an opportunity for some more fundamental alterations to the system. An important one of these is the mechanism that is used for expressing the system's linguistic rules. In the past rule sets have been written as pieces of code. Although this allowed for a great amount of control in the rule-writing process it had obvious limitations, *e.g.*, having to recompile after each change, and the writers of the rules requiring the knowledge to translate them into pieces of code. To avoid these problems a number of engines are now being implemented that are able to process sets of linguistic rules that have been written in a more appropriate language. Linguists can now write and test rules without the need to be able to understand the underlying engine's code. The programmers can also now concentrate on optimising the engines which process these rule sets.

Perhaps the clearest example of such an engine is the parser. In the past grammatical rules were added as pieces of code and the system re-compiled. This relatively inefficient grammar development process has now been superceded by ones in which the running system loads the grammar from appropriate files. This allows the grammarian to concentrate their effort on the grammatical rules and not the implementation of them. Using C++ the programmers have also been able to develop some very low level optimisations that have greatly increased the speed of the parser whilst at the same time reducing its memory requirements. The new parser is estimated to be some 10 times faster and require a fifth less memory. (This parser was not available in time for the MUC-7 evaluations.)

The developers view this re-engineering process as a natural progression of LOLITA from a research arena to that of the commercial world.

Addition of Dictionary Definitions to Knowledge Base

A natural language processing system requires knowledge at a number of important levels. The required knowledge includes:

Grammatical word information

knowledge about the structure of a word.

Semantic word information

knowledge about the meaning of a word, *e.g.*,
– `soccer' is a game played on a pitch by two teams,
– `sell' involves a transfer of money from a buyer to a seller.

World knowledge

knowledge such as:
– what particular objects are used for,
– why particular events happen.

It is widely recognised that knowledge about words and their meanings (the first two types) is already available in conventional dictionaries. However, these are aimed at human readers. For a computer to utilise knowledge in a dictionary the definitions need to be processed to extract and represent the information in a suitable form. A major project is currently underway which uses LOLITA to help process dictionary knowledge for computer use. The dictionary which has been selected for this project is the Cambridge International Dictionary of English (CIDE). The aim is to incorporate the knowledge contained in CIDE into SemNet. It is estimated that this will increase the size of SemNet from over 100,000 nodes to well over a million.

To be able to carry out this process LOLITA is used to analyse as much of the definition as possible. However, help is required in resolving various problems and ambiguities that occur in the original definition. This help is given by the user in the form of a question-answering session (see [10] for more details). The questions fall into a number of different categories:

choosing grammatical categories,
picking word meanings,
entering word information,
solving structural ambiguities,
finding referents for mentioned words,
finding referents for implicit objects,
making objects more specific,
naming events and entities,
naming relationships,
confirming the analysis.

The categories of questions in the above list also represents a rough estimate as to the order of the question-answering process (earlier questions such as the picking of word meanings may also occur in a different context later in the process). In practice the interaction with the natural language system is via a Graphical User Interface. LOLITA processes as much of the definition as it is able and then typically presents the user with a question and a list of possible answers. The user then simply uses the mouse to select the most appropriate answer. Occasionally a question is asked that requires the name of an entity (or relationship) to be entered; a text entry box is then provided for that purpose. It is anticipated that it will take several person years to completely analyse the dictionary. The system has been designed so that it can be used, with little in the way of training, by people who are not specialists in the area of NL. This greatly increases the number of people that can be used to enter the knowledge and so will reduce the amount of time that will be required to complete this project.

CONCLUSIONS

The LOLITA system is a natural language processor that has a core functionality on which a number of different applications can be built. The system was not designed to compete solely in the MUC tasks and in fact the MUC specific code forms a very small part of the whole. The first MUC evaluation to which LOLITA was applied was MUC-6. Since then the system's core has been substantially improved and this has lead to a significant improvement in the system's performance for the MUC-7 evaluation. We are pleased with this improvement and believe that this further validates our method of development.

Although pleased with the results there is certainly room for improvement. In particular approximately 20% of parses failed in the final evaluation. This had a substantial impact on the system's performance as all of the task applications rely on a `good' core analysis of which parsing is a key component. Although some recovery strategies were used, these had a limited effect. (Little effort was expanded in attempting to reduce the number of parse failures as a new parser is under development.) Of particular encouragement were the articles for which the system's underlying analysis was correct. In such cases the system was able to apply the higher level semantic and pragmatic rules to good effect. The system also suffered a drop in scores due to the lack of data that was available. The issues of more robust parsing and data improvement are currently being addressed. Major projects are underway to re-engineer large parts of the system (the parser being one), and to add a large amount of data in the form of dictionary definitions.

We feel that the MUC series of evaluations has been very useful to the development of the LOLITA system and has allowed us to focus development on some key areas of the system's core. Now that MUC has reached an end we shall continue to devote our main development effort to the core issues, in particular those mentioned above, i.e., re-engineering of the system, and addition of dictionary definition data. This engineering work, carried out by a university spin-off company, 3F Ltd, will allow us to reach our goal of bringing to the market, a number of sophisticated and powerful natural language processing applications based on an underlying core system.

REFERENCES

[1] Morgan, R.G., *et al.*, University of Durham: Description of the LOLITA system as used in MUC-6, In *Proceedings of the Message Understanding Conference (MUC-6)*, 1995.

[2] Smith, M.H., Natural Language Generation in the LOLITA system: An Engineering Approach, PhD Thesis, Department of Computer Science, University of Durham, 1995.

[3] Miller, G., Wordnet: An online lexical database, *International Journal of Lexicography*, 3(4), 1990.

[4] Long, D. and Garigliano, R., Reasoning by Analogy and Causality: A Model and Application, Ellis Horwood, 1994.

[5] Tomita, M., Efficient Parsing of NL: A Fast Algorithm for Practical Systems, *Kluwer Academic Publishers*, Boston, Ma, 1986.

[6] Wilks, Y., Preference Semantics, *Formal Semantics of Natural Language*, pp. 329--348, Cambridge University Press, 1975.

[7] Grosz, B.J., Joshi, A.K. and Weinstein, S., Towards a computational theory of discourse interpretation, *Computational Linguistics*, 21(2), 1986/1995.

[8] Hudak, P., Peyton Jones, S. and Wadler, P., Report on the functional programming language Haskell, Version 1.2, March 1992, URL ftp://ftp.dcs.gla.ac.uk/pub/haskell/report

[9] Brill, E., Some advances in rule-based part of speech tagging, *AAAI-94*, 1994.

[10] Poria, S., A Natural Language Engineering Approach to Knowledge Acquisition by the Analysis of Dictionary Definitions, PhD Thesis, Department of Computer Science, University of Durham, forthcoming.