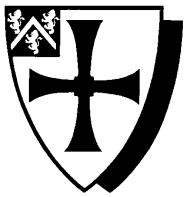
University of Durham



An Evaluation of LOLITA and Related Natural Language Processing Systems

Paul Callaghan

Laboratory for Natural Language Engineering,
Department of Computer Science.

Submitted in partial fulfilment of the
requirements for the degree of

Doctor of Philosophy

Abstract

An Evaluation of LOLITA and related Natural Language Processing Systems

Paul Callaghan

Submitted to the University of Durham for the degree of Ph.D., August 1997

This research addresses the question, "how do we evaluate systems like LOLITA?" LOLITA is the Natural Language Processing (NLP) system under development at the University of Durham. It is intended as a platform for building NL applications. We are therefore interested in questions of evaluation for such general NLP systems. The thesis has two parts.

The first, and main, part concerns the participation of LOLITA in the Sixth Message Understanding Conference (MUC-6). The MUC-relevant portion of LOLITA is described in detail. The adaptation of LOLITA for MUC-6 is discussed, including work undertaken by the author. Performance on a specimen article is analysed qualitatively, and in detail, with anonymous comparisons to competitors' output. We also examine current LOLITA performance. A template comparison tool was implemented to aid these analyses.

The overall scores are then considered. A methodology for analysis is discussed, and a comparison made with current scores. The comparison tool is used to analyse how systems performed relative to each-other. One method, Correctness Analysis, was particularly interesting. It provides a characterisation of task difficulty, and indicates how systems approached a task.

Finally, MUC-6 is analysed. In particular, we consider the methodology and ways of interpreting the results. Several criticisms of MUC-6 are made, along with suggestions for future MUC-style events.

The second part considers evaluation from the point of view of general systems. A literature review shows a lack of serious work on this aspect of evaluation. A first principles discussion of evaluation, starting from a view of NL systems as a particular kind of software, raises several interesting points for single task evaluation. No evaluations could be suggested for general systems; their value was seen as primarily economic. That is, we are unable to analyse their linguistic capability directly.

Declaration

The material contained within this thesis has not previously been submitted for a degree at the University of Durham or any other university. The research reported within this thesis has been conducted by the author unless indicated otherwise.

Copyright Notice

Acknowledgements

First thanks to the Engineering and Physical Science Research Council, and the Defence Research Agency at Malvern for funding my work.

Many thanks to those who have helped this thesis on its way, and to those who have made my time whilst working on it so interesting and enjoyable. You know who you are!

Particular thanks must go to Roberto Garigliano – the driving force behind LOLITA, and to Rick Morgan, my supervisor. It has been a pleasure to work with you.

Finally, I would like to thank my parents, Chris and Tina, for everything.

It doesn't matter if a cat is black or white, as long as it catches mice.

Deng Xiaoping, 1962

Contents

List of Figures

Note: Some figures may be difficult to read at thesis page size. Postscript for larger versions can be obtained from the author, P.C.Callaghan@durham.ac.uk.

Notes and Abbreviations

- &c = *et cetera*
- CEO = Chief Executive Officer (occurs in some MUC articles)
- CO, Coref, coref = Coreference task
- eg = *exempli gratia*
- GSJ = [Galliers and Sparck Jones, 1993]
- GUI = Graphical User Interface
- IA = Inter-annotator agreement
- IQR = Inter-quartile range
- ie = *id est*
- LOLITA = Large-scale, Object-based, Linguistic Interactor, Translator and Analyser. The NLP system being developed in Durham.
- MT = Machine Translation
- MUC = Message Understanding Conference
- MUC-6 = The Sixth MUC
- NE = Named Entity task
- NL = Natural Language
- NLE = NL Engineering
- NLP = NL Processing
- S&LP = Speech and Language Processing
- SGML = Standard Generalised Markup Language
- ST = Scenario Template task
- TE = Template Elements task

The words 'he', 'his', &c are used to refer to an anonymous individual, hence do not presuppose either gender.

Chapter 1

Introduction

This research attempts to answer the question: "how should we evaluate LOLITA?" We initially adopt this intuitive definition of evaluation: "to *evaluate* is to determine what something is worth to somebody" ([EAGLES, 1995] p15). LOLITA (Large-scale, Object-based, Linguistic Interactor, Translator and Analyser) is the Natural Language Processing (NLP) system being developed at the Laboratory for Natural Language Engineering (LNLE) of the University of Durham. LOLITA may be loosely characterised as a large-scale system, based on a rich semantics, which is intended as a base for implementing general NLP tasks in any domain. Our work will apply to similar systems, which for convenience we shall call "LOLITA-like".

We approach the problem in two ways. The first is *empirical*, reporting on LOLITA's recent participation in the DARPA Sixth Message Understanding Conference (MUC-6) [DARPA, 1995], then analysing LOLITA's results and MUC-6 itself. The second is *theoretical*, considering the implications for evaluation of LOLITA from the complementary viewpoints of science and software. This part will draw upon our MUC-6 experiences for evidence.

First, some important issues must be discussed: namely the background philosophy for the work and the key assumptions being made. This component, the "Methodology", is essential in a field where practitioners have diverse backgrounds (eg linguistics, logic, computer science), and hence have diverse views on what the goals of NLP should be, and diverse views on what constitutes proof, adequacy &c. It will provide a frame for the arguments of this thesis, and for discussing the *value* of the work. The "Methodological Criteria" for success are then outlined. Relevant 'Terminology' is considered next. After a discussion of the research context (or environment) for this thesis – which supplies further assumptions – the global plan of the thesis is presented.

1.1 Methodological Comments

In this section, the methodology for the thesis is outlined. The "Setting of the Work" considers the nature of this work. Since science itself lacks a universally accepted definition, what we mean by science is outlined in "The Neo-pragmatic View of Science". This is not intended as a universal definition of science, but as an explicit statement of our viewpoint as it affects this work. The paradigm of Natural Language Engineering (NLE) is currently receiving a lot of attention. The final section discusses the standing of this thesis to NLE.

1.1.1 Setting of the Work

The work is concerned with evaluation in the research field which attempts to build (software) systems to usefully manipulate Natural Language. In particular, it is concerned with work that is intended to supply a base system which can be used with many tasks in many domains.

Some may call this field 'NLP', but in our opinion, that term is used to cover a much wider

enterprise – such as the investigation of linguistic theories by computational means. Though useful in its own right, we do not see this wider research as directly relevant to our problem. Others may suggest 'NLE', but this is too narrow for our purposes. As discussed in section 1.1.3, NLE can be seen as a pragmatic approach to the field. How we evaluate the results of the field does not have a necessary connection with how work in the field is conducted, though there may be some influence. Hence, our work is not exclusively NLE or NLP. It is methodological: it is about them.

For convenience, we shall use the term NLP, but will bear in mind the kind of system envisaged (LOLITA-like), and the practical way in which we want to produce such a system.

1.1.2 The Neo-pragmatic View of Science

Surprisingly, to the uninitiated, there is little consensus on what is meant by 'science'[1], thus it is necessary to indicate what one means if one is to appeal to science in argument.

In this work, a neo-pragmatic view of science is adopted [Garigliano, 1996]. Science, Engineering, and Technology are seen as closely related endeavours, with the common aim of solving problems and hence allowing manipulation of the environment. Other ideologies have similar aims, but none are as successful as science: it produces results. It is supposed that the reason for this is the use of a collection of techniques which have been developed through history.

Neo-pragmatism does not suppose that science is about 'truth' (whatever truth may be): instead, it focusses on the usefulness of ideas, or more formally, on the usefulness of hypotheses. Science is thus about successfully discriminating between competing hypotheses. If one hypothesis about some phenomena allows greater control of that phenomena than another – perhaps it matches observations more precisely, or has predicted new phenomena which have been subsequently observed – then it is to be preferred over the 'weaker' hypothesis.

In particular, [Garigliano, 1996] views science as "an evolving and progressively *more* efficient method for acquiring an increased control over reality. Control here is defined, going from the weakest to the strongest, as predictive ability (that is ability to predict what external phenomenon will happen), modifying ability (that is the ability to interfere in a controlled, specified way with external phenomenon) and generative ability (that is ability to create things and processes that did not exist before)."

Notwithstanding, science is not perfect. There is no certainty as there is in logical derivations. Use of scientific technique is just an heuristic - a strategy which works with varying degrees of success.

1.1.3 Natural Language Engineering

The identification and adoption of the paradigm of Natural Language Engineering (NLE) [Boguraev *et al.*, 1995], a pragmatic approach to NLP, is a promising step in solving the NL system problem. It is a pragmatic view of *current* NLP: it attempts to direct contemporary work towards the medium-term production of useful NL tools. Several important characteristics of systems and of the methods used to develop them are identified, such as scale (the linguistic complexity of a system), efficiency, and use of appropriate techniques and tools. An advantage of NLE is to show a clear difference between general NL work and the practical sub-part of NLP, and to state the aims of that sub-part. It is not a 'cut down' version of NLP, nor a substitute for some notion of 'proper' or 'theoretical' NLP. Instead, it represents a large part of the motivations that gave rise to the NLP field: of building useful systems.

Evaluation is promoted as an important part of NLE. From the pragmatic viewpoint, evaluation in the market-place is the "only real testing ground" [Boguraev *et al.*, 1995]. As this is obviously impractical for research work, competitive evaluations and standard test sets are

[1] Consider the field of the Philosophy of Science, which attempts to arrive at such a consensus and to provide good methods of doing science.

suggested as intermediate alternatives. However, NLE does not in itself prescribe methods for evaluation: evaluation is essentially a good intention rather than a strict requirement.

As noted above, there is not a necessary connection between the evaluation and the method of work, so this thesis is not exclusively NLE: it develops ideas that can be used inside NLE.

1.2 Methodological Criteria for Success

These are problem independent criteria. Their purpose is to set standards for the way in which the work in this thesis is conducted and justified, especially with reference to the above argument about the need for rigour in the field. Our problem is itself methodological: problem-specific criteria will then be methodological criteria, so there will be some repetition between this section and the corresponding part of the next chapter. We should also consider the limitations of the thesis form, to set realistic criteria.

As noted before, the thesis has two main parts: an empirical part which examines the participation of LOLITA in MUC-6, and a theoretical part which considers how one should or could evaluate LOLITA. We consider criteria for each part separately. For the empirical part:

- The main requirement here is thoroughness. MUC-6 has already been run, so our only option is post-event analysis of the available information.
- Because of the scale and nature of the event, it is not possible to change some of the conditions and re-run the competition.
- However, we should ensure that any changes in the setup we suggest are feasible, ie they should be well supported by evidence or argument.

For the theoretical part:

- We should show a contribution to the wider problem. A full solution is unlikely in the context of a thesis, especially for evaluation in NL, which is, as we shall see, a hard problem.
- We should show that the ideas suggested are 'scientific' under the neo-pragmatic viewpoint. That is, they should lead to an improvement of the situation, or increased control in our underlying problem.
- Suggestions should be implementable – ie, not vague proposals which need a lot more work before they can be applied. We do not see much value in vague suggestions for evaluation work, where the real detail is not adequately considered.

1.3 Terminological Issues

This section introduces some terms which are important to the main argument. We shall not need much terminology until chapter 8, so we shall discuss only the general terms of Evaluation and Assessment.

So far, we have been using an informal notion of Evaluation. [Pallett and Fourcin, 1996] argues that we should distinguish between Evaluation and Assessment. Evaluation is defined as the direct calculation of something. Assessment is defined as the estimation of something. Assessment is the standard term used by the speech community.

Evaluation is the strongest method, providing the best answers. If one cannot calculate a value directly, then estimation (assessment) is a possibility. But, there are limits to estimation. Estimations rely on assumptions which must be believable if the estimate is to be accepted. There are often many plausible assumptions possible. Different levels of assumption result in differently plausible answers.

The plausibility of an assumption is usually dependent on the context of use. If the assumptions required for an estimation are unacceptable in a context of use, then estimation is no longer safe. So, even estimation is not possible for some quantities.

To avoid tedium by saying Assessment and/or Evaluation, or to avoid the need for inventing a term which subsumes both, we will use 'evaluation' in informal discussion in the text up to chapter 8.

The following, seemingly irrelevant, example is provided to show the complexity possible in a simple Evaluation or Assessment.

The leafiness of a tree is evaluable by counting leaves, albeit a long and repetitive process. A less exact answer may be obtained by making various assumptions about the distribution of leaves in a tree and the volume of the leafed part. Evaluating this volume introduces a new complexity. It is not a discrete quantity like the number of leaves, nor can it be measured in a simple way. But let us assume that it has been determined accurately. Then the simplest way to estimate leafiness is to assume a constant density of leaves, determined by counting the leaves in a chosen volume, and to multiply to the tree volume. This requires assumptions about the uniformity of leaf distribution, and that the sample volume is representative of the whole tree. There are no guidelines for choosing the sample volume, so this can be any size.

Other options are possible – the leaf density in certain regions of a tree (such as the centre, or the extremities) could be established. In other words, there are a large number of assumptions which could be used, leading to answers of differing exactness. Obtaining better answers requires stronger assumptions (ones that are more believable) and more work.

But, exact answers are not essential in all contexts. The exactness required of an answer is usually related to the consequences of a bad answer compared to the effort required to obtain a good answer.

A tree grower may need a leaf count to gauge how much pesticide to use. A bad answer may lead to the tree suffering and the grower losing money over it. A good answer may protect the tree but take a lot of the grower's time to obtain. Thus, the grower has to find a tradeoff between exactness of answer and the effort taken to obtain it. Additionally, the grower may arrive at a good estimation method by using different techniques on several trees over time. Thus, he trades probable damage to several trees for the knowledge of a good estimation technique. Should he trade many trees to obtain a good answer in a short time, or fewer trees to arrive at an answer over a longer time?

A further complication: the leaf count will change over time, new leaves appearing and others dying. An exact answer will only be accurate for a short time, and assessments will gradually lose validity.

To summarise: Evaluation is exact or precise, and assessment is an estimate where evaluation not possible. Assessment a weaker form of evaluation. Simple evaluations can be used towards assessment of the whole. The worth of estimates depends on the assumptions required in the context of use. Essentially, there is a tradeoff of exactness against the effort to obtain an answer. There are no strict guidelines on how to manage a tradeoff, but some information may be obtained by preliminary experimentation. The amount of experimentation is again a tradeoff for information against the consequences of not experimenting. Some things may not be assessable – where the assumptions required are not believable in the context, or the context requires an exact answer.

1.4 Research Context

We outline the context of the work, discuss current internal testing and evaluation practice, and examine the assumptions inherent in the system and the LNLE. These assumptions are relevant to the argument of the thesis.

1.4.1 History and Introduction

The LNLE is engaged in the development of the LOLITA system. Work began on this in 1986 when Dr. Roberto Garigliano arrived in Durham. The group now includes several lecturers, support staff, and PhD students. LOLITA is designed to be a large-scale and general NL system: fundamental language-based facilities form a 'core' system, which is then used in building specific applications. This core includes syntactic analysis, semantic analysis [Baring-Gould, 1997], NL generation [Smith, 1996], and reasoning [Shiu, 1997]. Underlying this core is a large knowledge base encoded in a representation language developed in the LNLE; the 'Semantic Net' contains some 100,000 nodes. More details on the working of the core (as relevant to MUC-6) can be found in chapter 3.

Example applications of LOLITA include simple meaning-based translation, contents scanning [Garigliano *et al.*, 1993], implementation of a plausible reasoning model [Long and Garigliano, 1994], and Chinese language tutoring [Wang, 1994]. Ongoing and future work includes connexion to speech input [Collingham, 1995] and output [Callaghan *et al.*, 1994], and handling of metaphor [Heitz, 1996]. Currently, the core of LOLITA uses only symbolic techniques, but some work on adaptive techniques has been done [Nettleton, 1995], and wider use of such techniques is being considered.

In a system of LOLITA's size and goals, evaluation is of great importance. Not only does the basic functioning of the system need to be appraised, but more abstract concepts such as the architecture and the theories implemented within this architecture require some kind of assessment. However, in such a system it is also important not to evaluate incorrectly.

1.4.2 Current Internal Evaluation and Testing in LOLITA

In this subsection, the author is presenting LOLITA group policy on testing and evaluation, namely the methods currently used to assess performance and determine correct functioning. It is not an endorsement of such policy. One important point is of group structure. The sub-groups each have their own standards and methods for testing:

- One or two people work actively on the rules inside LOLITA, which includes the net data and grammar. This is a small group because of the knowledge-intensiveness of such work on the system. For example, in common with many systems, the grammar is finely balanced and performance often degrades if altered by a non-expert in the style and conventions used in it.

  Testing is mainly a check that the intended result is obtained without degrading the usual performance. Only recently has such regression testing been required before *any* change is permitted on the source code or data (the possibility of automation and the consequences of *not* doing such testing promoted the necessity); before, it was done occasionally. Basic testing is on a few straightforward examples (currently around 100 tests), checking most stages of analysis from parsing through to the Coref task on simple texts, with a failure being a deviation from expected output or a big increase in resource usage.

- Research students usually work on specific areas of the system, such as utilising causality in analysis. Their work is not usually added to the system until late in their research, and is not always made part of the standard analysis functionality, so it does not affect the main system. Much of their evaluation is done using small texts which are known to be handled well by LOLITA: so, correct analysis results are assumed. This means that their algorithms are tested on reliable input.

- Several people work on the general technical aspects of the system. The author's main experience with the system falls in this category. For this group, the main criteria in work is to preserve output performance on test sets such as our MUC-6 development sets. One aspect of the work involves setting up new abstractions for the developers above to use. This is usually tested by small test sets supplied by the developer concerned, indicating expected performance. This aspect includes new 'languages' in which the developers express analysis

rules (such as the pragmatic type checks), changes to the deep working of code (such as preserving original text), and supplying code to manipulate analysis entities or results (such as search algorithms). Another aspect is of efficiency. Section 3.3, which presents the author's contribution to the MUC-6 preparation work, will show how important this is in LOLITA. Another criterion is improved run-time characteristics over arbitrary texts. Improvements in performance are obviously welcomed: differences in output performance are manually checked by an expert to determine if the change is an improvement, before the changes are made permanent.

Additional protection was provided during MUC-6 development by regression testing on a small set of articles. This was also the set used for rule development – so it was used as a gauge of progress. The main criteria was an increase in scores, although this has allowed changes in *how* marks were obtained to go undetected. Thus, lost marks in one area could be balanced by a big improvement in another area to show an overall small improvement. At that time, we did not have good tools to quickly check qualitative differences in output (see Appendix B). A form of this MUC-based regression testing is still done, months after the conference.

1.4.3 Informal Discussion on Evaluating LOLITA

Clearly, these approaches are very informal. Several questions arise: are these approaches adequate and/or acceptable? This discussion is a prelude to chapter 8.

A first point is, what do we require from testing? There is no firm statement of requirements for LOLITA, or of what a LOLITA analysis should be like (apart from maybe the code itself), so we have no set goals to test against. We are not trying to test some theory of (computational) linguistics (such as building a system with a certain semantic model) or to test some design goal (eg of using unification as the main mode of computation), so we cannot derive criteria from such goals. In a weak sense, our work does test the NLE claim that useful systems can be built using 'Engineering' principles (as opposed to the more conventional 'Scientific' principles of working with a single, coherent theory).

The main requirement in internal performance is that it gives good results in applications. This is a very pragmatic view, and is in accord with the NLE orientation of the group. Unfortunately, defining this requirement more precisely is difficult.

Specific metrics such as for parsing are not useful under this view (see section 7.2). They take account of detail which is often irrelevant to most tasks. Furthermore, metrics have only been firmly established for the *earlier* stages of NL analysis, with little work known for the more complex stages of semantics and above. Concentrating on the quantifiable aspects could introduce unacceptable biases in our work.

This lack of firm criteria is mirrored by LOLITA's long term goals and development history: there is a general idea of how we would like LOLITA to be, and development consists of attempting to move in that direction. There is no detailed plan of stages. The complexity of such NLP systems and novelty of research in those areas hinders such planning. There is a concept of a "minimum system", which is effectively the basic architecture for LOLITA, characterised by the extensive use of the semantic net. It could be viewed as the framework into which all work must fit, and to which all work should add. Extensions are often driven by contemporary requirements (eg MUC-6), but limited by available resources. Our open-ended development is very much related to the lack of basis on a particular theory: we add pieces when they seem to be required by the applications we are currently working on. This includes adoption of techniques, such as a 'pre-parser' for NE expressions, that are successful in a task of interest.

LOLITA is very much a system in development: it is certain that we've not achieved the stage of a perfect or complete minimum system. We are not sure how far we have got, or how far we have still to go. Furthermore, work is continuous and the position changes all the time. In such circumstances, it does not make much sense to evaluate the system as if it were a product. What is needed is a strong diagnostic investigation.

To conclude: in LOLITA's current state, the informal methods used above are probably

adequate. But we would like to use LOLITA in a wider context and to compare our work to others'. So, a more thorough approach is needed.

1.4.4 Assumptions in the system

There are several assumptions involved when working with such a system as LOLITA. Some of these may be controversial for some people in the NL field. Certainly, we note that these kinds of assumption do not get much mention in the literature. That is, a lot of the literature is built on a different set of assumptions which are often incompatible with ours.

- Our first, and major, assumption is that it is possible and useful to produce a large-scale NL processing system that is not limited to any particular task or domain.
- That there is substance to the intuitive notion of the worth of such a system without reference to particular tasks. That is, the worth of the 'core' language facility can be measured in some way.
- Furthermore, that we can also measure progress in building such a system.

1.5 Structure of the Thesis

Chapter 1: Introduction has just introduced the background philosophy, outlined methodological criteria, and presented the research context for the work. The problem addressed in the thesis is described in **Chapter 2: Statement of Problem**.

Four chapters comprise the practical section of this thesis, on the LNLE's participation in MUC-6. **Chapter 3** describes the work done to prepare LOLITA for MUC-6, and explains the parts of the system relevant to MUC-6. The author contributed significantly to this work. LOLITA's results in MUC-6 are covered in two chapters. **Chapter 4** examines LOLITA's task performance on a specimen article (given in Appendix A): this is a concrete introduction to the MUC-6 tasks, an explanation of how LOLITA obtained particular output, and also an examination of what is involved in performing the tasks. **Chapter 5** presents the overall scores, examines features of these, and compares original performance with current performance. It also investigates LOLITA performance relative to other competitors in MUC-6. The final MUC-6 chapter, **Chapter 6**, examines the details of the evaluation itself, the tasks and methodology, and suggests changes and additions to the format.

The theoretical component consists of a literature review, **Chapter 7: Related Work**, and **Chapter 8: Evaluating LOLITA-like Systems** which considers what evaluation can mean for a system like LOLITA. The achievements of the thesis are outlined in **Chapter 9: Conclusions**, comparing the results against the methodological and problem-specific criteria, then drawing overall conclusions and making suggestions for further work.

Chapter 2

Statement of Problem

This chapter identifies the problem-specific aims of this thesis and outlines the criteria for their achievement. Remember that the problem is methodological in nature, so there will be some overlap with the methodological criteria (section 1.2). The aims are presented in the order covered in the thesis.

2.1 Aim 1: To Participate in MUC-6, and to Analyse the Results of MUC-6

As indicated in section 1.4.2, LOLITA had not been formally evaluated in an externally recognised way. This was recognised as an omission which needed rectifying. MUC-6 was seen as the evaluation most relevant to our work, and also as one which was well recognised and held considerable prestige.

Though reservations existed about MUC-6 within the LNLE group, especially as the set of tasks was narrowed down to exclude some of the ones which were of more interest to us, it was believed that the whole exercise would be beneficial. Hence we participated in MUC-6 as an *experiment*, both to gain experience about evaluation and about the process of evaluation, and to see how well we would do. This would be valuable evidence for any work we wanted to do in evaluation.

The preparation work and the evaluation has been done[1], which leaves the following sub-aims:

2.1.1 Aim 1.1 : LOLITA in MUC-6

- To provide detail about the workings of the LOLITA system used in the MUC-6 evaluation, extending the brief details given in the MUC-6 conference paper [Morgan *et al.*, 1995], and to describe the work done in preparing LOLITA for the evaluation, a significant part of which was done by the author. This description of work done is interesting as an indication of LOLITA's adaptability and maintainability, both important criteria for NL systems.
- To provide a more detailed analysis of LOLITA's performance. This was insufficiently done in the proceedings paper – both in terms of successes and of failures. The proceedings paper was insufficient because of lack of space (in the paper), lack of time (considering the scale of such an analysis), and lack of experience (in performing such an analysis usefully and to a good depth). We will aim to find new ways of looking at the MUC-6 results. We will compare our performance to that of other participants, where possible.

[1] The MUC-6 evaluation was run in the first week of October 1995.

- To consider the relevance of the MUC-6 results to the LOLITA project. LOLITA was not expressly designed, as other systems were, to perform the kinds of tasks used. Do our results show that we lose marks through lack of specialisation? Or because of some other factors?

Criteria for Aim 1.1

The descriptive component of this aim has no definite criteria, apart from presenting information of interest about the LOLITA system and its workings which were previously unpublished, or published in limited form.

The analysis component will give a more complete picture of LOLITA's performance in the MUC-6 tasks. This thesis rectifies the problems noted with the proceedings paper. As well as the successes, we will examine the errors – especially for the points where we expected to do well. Our novel analysis methods will be successful if the information is interesting and useful – if it shows other aspects of the MUC-6 results.

The question of relevance is more open: this cannot be more than a discussion of relevant points. The question can also be discussed as part of the theoretical framework.

2.1.2 Aim 1.2 : Analysis of MUC-6

What do the results of MUC-6 mean? Clearly, MUC-6 is not an absolute standard, so MUC-6 itself requires some evaluation. To the best of our knowledge, no detailed, independent discussion of MUC-6 has been published, so we aim to supply one viewpoint. Several aspects can be investigated:

- The framework: the goals of MUC-6, the changes to these as they were implemented, the results with respect to the original goals.
- Task designs: output format, definition of task.
- The general methodology: experimental design, the competition format, scoring methods.
- Significance of its results.
- Suggestions for future MUC-style evaluations.

Criteria for Aim 1.2

Such a critical effort is open-ended, so precise criteria are not possible. Again, the criteria mirror the methodological points: desirable attributes of our analysis are thoroughness, detail, objectivity. Good criticism will suggest improvements and modifications to the MUC formula. The suggestions must be implementable, or at least feasible and well-justified.

2.2 Aim 2: Theoretical Basis

To investigate what evaluation of general NLP systems can mean. The literature review will show that general NLP systems have been relatively neglected in evaluation work, so there is a place for a "first principles" investigation. We consider the conditions and restrictions inherent in such systems.

Criteria for Aim 2

Since this will be an initial investigation, and hence open-ended, criteria are elusive. We suggest the following as ideal criteria, some of which may be unachievable.

- Contribution to the literature: a good discussion of the aspects of general NLP systems as they affect evaluation. This will cover topics such as the underlying assumptions, the design, the practical details, and the logistics, based on our experience with LOLITA.
- Consideration of what kind of evaluation can be done on such systems.
- Examination of the *notion* of such systems. Are they a good idea? Or impractical and inherently limited?
- Suggestions for actual evaluations. We can set several sub-criteria for these. A concrete, implementable, and feasible proposal would be ideal. At the very least, suggestions should be relevant to current and existing work or systems. The results should be pragmatically useful.

Chapter 3

The LOLITA-based Applications used in MUC-6

This chapter describes the system of October 1995 as used in MUC-6, and the work done on LOLITA to allow participation in MUC-6. The chapter begins with a discussion of Haskell, the main implementation language in LOLITA. This language is relatively unknown in NL (and AI) work, a situation we hope the discussion may contribute to changing. Some knowledge of it is needed in the description of LOLITA and in the description of the thesis author's work.

Next, LOLITA is described, with more detail than was possible in [Morgan *et al.*, 1995], where space was limited. Particular attention is given to the MUC-relevant parts, such as the reference resolution algorithm, which were explained briefly in the paper and were the subject of questions at the MUC-6 conference. The work done on the system by the author is then presented. This is examined in some depth; it is the 'implementation' part of this thesis. It includes information about the maintenance and profiling of a large functional program. The chapter ends with conclusions about the adaption of LOLITA for MUC-6 and the work done for MUC-6.

3.1 Haskell: the main implementation language in LOLITA

Most of LOLITA is written in Haskell [Peterson and Hammond, 1997], a non-strict functional programming (FP) language[1]. The purpose of this section is to present basic information on Haskell which is relevant to subsequent discussion of LOLITA, and to raise interest in the language.

Of the main programming languages, LISP is its closest relative[2]. One key question when comparing programming languages is that of what the new language offers over existing languages. Haskell does not offer facilities which are missing from LISP, but it does make them more accessible and easier to use. The ideas behind Haskell have a stronger theoretical basis, and they promote some modern ideas about software engineering. The similarities between LISP and Haskell include, in no particular order:

- **Garbage-collected memory**: no explicit memory allocation or deallocation is done – it is done by the language's run-time system. The term 'heap' is used for the fixed-size memory pool on which garbage collection operates.
- **Based on the lambda calculus**: – a calculus of nameless functions ([Hindley and Seldin, 1986] is a good introduction). Function values can be built and manipulated directly, as

[1] We use the `ghc` compiler [Glasgow, 1997] for most work. We have also used the `hbc` compiler [Augusstson, 1996]. Other compilers are available.

[2] COMMON LISP [Steele, 1990] is assumed in this discussion.

"first class citizens". "Higher-order functions" are distinguished, as those which return, or accept as a parameter, a function value.

- **Programming with functions**: programming consists of writing functions. Functions can be built by combining other functions. Functions can be partially applied, that is, an n-ary function given m arguments ($m < n$) has meaning, and will return results when the remaining arguments are supplied. For example, `foo = map times2` defines a function which applies the doubling function to all numbers in a list.

  Programs are 'run' by *evaluating* some expression involving the main functions in the program. Note this use of 'evaluate' to mean calculation or computation – this is the sense understood in mathematics.

- **Recursion**: is the main control construct, with others such as loops being implementable using recursion.

But there are the following important differences in Haskell, again in no particular order of importance:

- **No Assignment**: Haskell has no notion of assignment, so a function's result is completely dependent on its arguments. This property is called *referential transparency*. The lambda calculus does not include assignment. Assignment to a shared run-time state (as in LISP) mean that side-effects are possible, with the consequence that function results do not depend solely on their arguments, but also on the shared state at the time of execution. This can complicate debugging since the dynamic state of a program must be considered. It can also reduce the possibility for optimisation by the compiler, since the side-effecting behaviour has to be preserved. Thus, more optimisations are possible for functional languages and may be implemented more easily [Santos, 1995]. With no side-effects, computations are also inherently parallel, easing transfer of a program to modern parallel machines [Trinder *et al.*, 1996].

- **Inherent Non-Strictness**: this means that a function's arguments do not have to be fully evaluated before a result can be returned. Thus, computations only need to be evaluated when required, and the compiler is free to rearrange evaluation order for optimisation. One consequence of non-strictness is that infinite or possibly undefined values can be described and manipulated naturally in the language without the baggage of allocating resources, or of programmer-supplied book-keeping.

  As an example, the list of all natural numbers can be expressed as `from 0` using the definition below. `[Int]` denotes a list of `Int`egers, and `:` is the 'cons' operation as in LISP. The infinite list of integers from n is n cons'd on to the list starting from $(n+1)$. Non-strictness means that the recursive call to `from` won't be done immediately (LISP would continually recurse until the stack was exhausted &c.), but it will get evaluated when a caller tries to use the list, or when it *demands* more values. The list of all prime numbers can be described in a similar way.

```
from :: Int -> [Int]
from n = n : from (n+1)

primes :: [Int]
primes = primes_aux 2
primes_aux n = n : filter (not_divisible_by n) (primes_aux (n+1))
               where not_divisible_by n m = m `mod` n /= 0
```

A related concept is *laziness*. *Full laziness* is the property of doing no computation whose result is unused, and of not repeating a calculation. Often, laziness trades time for space, so repeated calculations are traded against the space for storing the result for subsequent uses. Full laziness is probably impossible in the general case, since it requires extensive compiler effort to analyse code, and the benefits are not that great. Implementors of Haskell must

guarantee non-strictness, but the level of laziness provided is varied, often set to provide a trade-off with the optimisations (such as some compile-time evaluation). Compilers often allow the user to control the degree of laziness optimisation.

One use for laziness is illustrated below: `silly` calculates an expensive local value from its first argument *only*, and adds this to its second argument. If the (sub-)expression `silly 100` is used frequently in some piece of code, the repeated summing of 1 to 100 can be avoided by a simple compiler transformation.

```
silly :: Int -> Int -> Int
silly x y = sum_to_x + y
            where sum_to_x = sum [1..x]
```

It is possible to get similar effects to non-strictness/laziness in LISP with some extra work, but the point is, this facility is central to Haskell.

- **Typing**: Haskell has a superior type discipline. LISP prescribes few restrictions on the types of values, with much checking being done expensively and repetitively at run-time. Haskell does *all* type checking once only at compile-time, like imperative languages. But this does not mean loss of flexibility. For example, LISP allows lists of mixed type because there are few constraints on type. Haskell requires the elements of a particular list to be the same type[3], such as all `Int`egers. Other lists can have other base types. Mixed lists are possible by defining a new type which contains the types desired in a mixed list. The extra housekeeping for this is minimal.

  In the example below, the `::` introduces a "type signature" for a value. The compiler can infer these, but it is good practice to provide these for purposes of documentation and sometimes to force a certain *specific* type for a definition.

```
flip :: (a -> b -> c) -> b -> a -> c        -- 1
flip :: (a -> b -> c) -> (b -> a -> c)      -- 2
flip f b a = f a b

const :: a -> b -> a
const x y = x

constInt :: Int -> b -> Int
constInt i y = i
```

  The `->` indicates a function type, so `Int -> Int` is a function taking an `Int` and returning an `Int`. The `->` operator associates to the right: the example signatures for `flip` are equivalent. The lower case letters in the type signatures are 'type variables', which can be instantiated to any well-formed type. Where a type variable occurs more than once in a signature, all instantiations of that variable must unify. This parametrisation by type is called 'polymorphism'. For example, an `Int`eger and a type variable `a` are unifiable, whereas an `Int` and a `Char`acter are not.

  `flip` takes a 2-ary function and two arguments, and applies the function to the swapped arguments. `const` takes two arguments and returns the first. It does not matter what types the arguments are. Both functions are higher-order. An example: the expression `flip const 1 2` will evaluate to `2`. A more specific version is shown in `constInt`: the definition is the same, but the signature is more restrictive – it can only operate with an `Int` as the first argument, and hence returns an `Int`.

  The utility of polymorphism is illustrated by the expressions `flip const`, `flip (flip const)`, `flip (flip (flip const))` &c. The result type is `b` for odd counts of `flip`, `a` for even counts. Errors of mismatching types cannot occur, and errors in the source code

[3] More correctly, it requires the types of elements to be unifiable.

will be detected at compile time. Polymorphism is used in defining the list type, and the functions which operate on lists: the functions do not need to know what the elements are, so they can be any type, as long as their usage is consistent.

Additionally, functions and operators can be safely overloaded for families of types, called 'classes'. Prime examples are for equality and arithmetic. There is a notion of inheritance: for example, `Ord` is a family of types for which ordering predicates (eg `<=`) are defined. These can require the definition of equality `==` from the 'superclass' `Eq`. An `Ord` instance is only valid if it is also an instance of `Eq`. Note that this notion of class is based on type, and not on structure (as in C++). Users can define new classes and subclasses, and add their own types to existing classes by supplying appropriate class members.

- **Definition of new types**: Simple types such as numbers, characters, and boolean values are primitive in Haskell, as are the container types of lists and tuples. These can be arbitrarily combined to form lists of tuples or lists, and tuples containing tuples or lists. The programmer can supplement these with his own specific types, allowing trees, records, graphs &c. These new types are fully compatible with the type system. They don't have to be simulated as in LISP.

- **Pattern matching**: This helps syntax and simplicity by easy decomposition of, and naming pieces of, data structures. The example below shows the Haskell version of the LISP `cdr`. There are two cases – an empty list, which causes an error, or an element cons'd to another list (the required tail). However, the convenience of pattern matching does allow writing code at too low a level of abstraction [Peyton Jones, 1996].

```
tail :: [a] -> [a]
tail [] = error "tail of empty list"
tail (x:xs) = xs
```

- **Cleaner syntax**: This is a minor point, but the syntax is an improvement over LISP, and may be partly due to the style of definition of programs: small functions are combined with higher order functions to produce more powerful functions. There is much reliance on operator precedence and associativity, which reduces the number of parentheses needed; eg, `sum 1 10 + factorial 20` is understood as `(sum 1 10) + (factorial 20)` New operators may be easily defined, and can be safely overloaded by type. Layout conventions remove the need for braces and statement separators.

However, there are disadvantages:

- **Still a Research Language?**: Most of the points below can be subsumed under the question of whether Haskell is suitable (or ready) for widespread use. There are many open research problems. The compilation technology is still being developed. There are few automated tools. LOLITA is itself, in part, an experiment as the largest known non-compiler program in Haskell. Other significant uses of Haskell are listed in [Hogg, 1997].

  Haskell is definitely useful as a teaching language. Many universities teach functional programming to undergraduates. Even if the students do not continue to program in Haskell, the discipline and ideas should still benefit a programmer when using conventional languages.

  We note that some of the good ideas from functional languages are influencing mainstream work. Sun's Java [Sun, 1997] is one example: that language is garbage-collected, and some other aspects of functional languages have been incorporated into it by the Pizza project [Pizza, 1997].

- **Compilation Technology**: Current Haskell compilers are much slower than compilers for mainstream languages. They are usually written in Haskell, and are hence research projects themselves, with no strong guarantees of a high degree of correctness &c (though fixes are quite prompt and help is freely given). The executables produced are large, for reasons

such as trading code size and complexity for better speed, and the implementation of non-strictness. The basic LOLITA is around 12Mb on a sun4 computer, not including data. The software object files are together much larger because of linking information, symbol tables &c. Thus, there are currently great overheads of time and space in compiling. These have a serious effect on development work, especially since they slow down the debug-compile-test cycle.

- **Strictness problems**: In some pieces of code, the laziness or non-strictness can get in the way and cause inefficiencies. A common example is in large, frequently updated and used data structures. Here, an update is not fully evaluated until it is re-read, leaving closures (unevaluated expressions) in memory. This can represent a significant amount of memory, as well as additional time for evaluating whilst reading, compared to a completely evaluated (strict) update.

  The main compilers can produce strict versions of code, but this requires programmer intervention. At present, detecting problematic code is a (skilled) manual process, with little support from tools such as profilers. More research and automated tools are required.

- **State**: With no side-effects, algorithms which rely heavily on state or are naturally specified with state become awkward to code, or inefficient. Examples of these are graph algorithms, or in our case, efficient parsing.

  In general, some 'state' value must be passed around and operated on. It is straightforward to build abstractions in functional languages to hide this state-passing, such as monads [Wadler, 1995] or stream processors [Carlsson and Hallgren, 1993].

  However, this passing around of a single state value results in a limitation on evaluation sequences and on possible parallelism. A state cannot be read until the pending updates are evaluated ('forced') to determine whether they have an effect on the result of the read. Some relief is possible by dividing the read-only and read-write portions of the state, and providing abstractions which allow free passage of independent values.

- **Debugging and Profiling**: Ease of debugging is important, unfortunately. There are no widely-available tools yet for FP. Current methods are more or less the equivalent of manually placing print statements at key points, then finding and reading the results from `stderr` – a tedious operation. Since conversion of values to printable strings involves *evaluating* them, such debugging aids can significantly alter program behaviour. They can also cause recompilation when pragmatic details of a module interface change (the current compilers export some information about strictness in interfaces).

  Profiling tools are not well-established either. Many are still research projects, and can take some effort to apply, both in recompilation, determination of *how* to profile, and obtaining useful results.

  It is sometimes said that functions can be tested and debugged easily because of their simplicity and of referential transparency. But, in a complex program with many functions, the interaction between functions can be hard to analyse: a particular case is where an object is modified by successive layers of rules, where the action of higher layers depends on earlier layers. Put another way, it is still possible to write obscure code in FP.

- **Programmer intervention**: The danger is that seemingly innocent code can require unreasonable amounts of memory – called *space leaks*. There is a similar problem with time – for example, repeated evaluation of sub-expressions when a compiler cannot automatically identify opportunities for laziness.

  In general, the time and space behaviour of arbitrary functional programs is not well understood, and must be determined by experimentation. Often, different compilation strategies can have big effects on the eventual behaviour. No sophisticated tools exist to help in this task. Essentially, removing such leaks, when they are a problem, needs a lot of experience. The author removed several from LOLITA during his profiling work (see section 3.3.3).

- **Lack of specific NL tools**: The 'standard' languages of LISP and Prolog are established enough to have several widely available and well-established tools to help with NL and AI

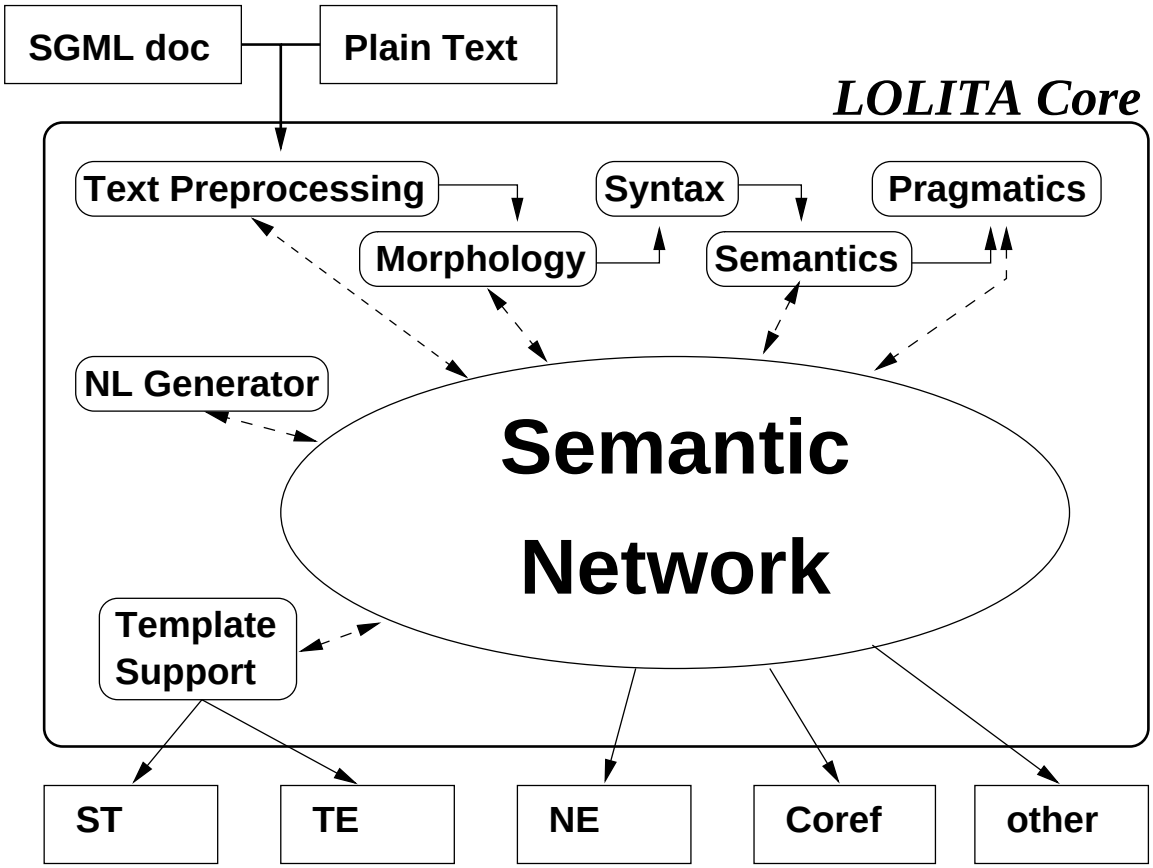


Figure 3.1: Block Diagram of the LOLITA Core plus some applications

work. Haskell would benefit from an implementation of such tools, or from the ability to interface easily with them.

To summarise, use of Haskell has its advantages and disadvantages. It is not widely used in NL work, though there is no reason why it should not be [Frost and Launchbury, 1989]. Our use of it has some research aims outside NL work. However, we are not really using the facilities of FP to best advantage – not aiming to produce showpiece code. For one thing, it would be too expensive to rewrite older portions of LOLITA when some new development was made: it is one of the largest functional programs in the world. But, we have pushed Haskell implementations: LOLITA is used as a test application for `ghc` compiler [Glasgow, 1997], and as a test for their parallel execution Haskell work [Trinder *et al.*, 1997].

3.2 Description of LOLITA as used in MUC-6

3.2.1 Overview

LOLITA is designed as a core system supplemented with a set of small applications, the former supplying basic NL facilities to the latter. Figure 3.1 shows the MUC-relevant parts. The most important part of the core is the large knowledge-base, which we call the Semantic Network, frequently abbreviated to 'net'. It is heavily used in most stages of analysis, with intermediate results being built in it. The final results of analysis are a disambiguated logical representation of the input. The analysis stages are fairly standard, and are arranged in a pipeline. Each stage is implemented in a rule-based way. We do not currently use any form of stochastic or adaptive techniques in the main system. Note that non-strictness means that 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.

Applications read the results of analysis from the net, and generally interrogate the contents of the net. Inference algorithms can be used to derive consequences of the analysis results or system knowledge. 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 net into English (or, recently added, into Spanish [Fernández, 1995]).

LOLITA occupies around 60,000 lines of Haskell, plus some 6,000 lines of C, in 300 modules. It can run on many machines (ie, those for which there is a Haskell compiler), and is normally given a heap size of 60 megabytes. To give an idea of speed, the final evaluation, if run on a single 70MHz SUN machine, took approximately 24 hours.

After some comments on the style of development on LOLITA during MUC preparation, the semantic net and its reasoning facilities are explained. The main analysis stages follow this, and finally detail on the implementation of the MUC-6 tasks.

3.2.2 Development Style and Effort

In line with the design philosophy of LOLITA, the MUC-6 tasks were implemented as *users* of the results of core analysis. For example, the basic coreference algorithm selects new concepts which are linked to more than one phrase in the text, subject to a few conditions. Development effort therefore concentrated on the core, although work was oriented towards those parts of the core most used in MUC-6. The addition of new functionality to the core was a significant part of the work. There were opportunities for implementing elements of a specific task without modifying the core, possibly resulting in initially better scores, but these were avoided:

- In most cases, the core would benefit from the general approach – other applications could make use of the new functionality.
- The 'fixes' often relied on weak assumptions about the core's contemporary inadequacies. Obviously, these cannot be relied upon.
- There would be debugging and maintenance problems when the core's treatment was improved.
- Experience supported this view.

Work had three main aims: implementing the tasks, ensuring that the core analysis was reasonable, and making the system more efficient so that the bottlenecks in testing could be eased. The author was mainly concerned with the latter, followed by the task functionality. Basic versions of Named Entity and Coreference were implemented for the MUC-6 dry-run. The dry-run performance was taken as a baseline, and the dual approach of increasing scores by adding functionality and by correcting the core analysis was begun. A subset of the dry-run texts was used in a nightly evaluation to check that performance had not deteriorated throughout the previous day's work. A second test set of non-MUC-6 texts were also tested to ensure that LOLITA's general functionality was not affected. As the speed of LOLITA improved, and as new test data was released, the MUC-6 test set was changed and expanded in size. Modifications to the system were not 'checked in' (under the Revision Control System – RCS) if scores had deteriorated, unless the fall in scores was shown to have removed a quirk that produced apparently correct results for the wrong reason.

Approximately 35 person-months was spent preparing the system for MUC, at least a quarter of which is due to the author. Only a small part of this time was spent on code specific to MUC-6 and not useful to the general core or to future applications (estimate: 5%). Some work did not contribute to the final system, such as experimenting with Brill's tagger [Brill, 1995], or with tables of data (the MUC gazetteer and a large list of common company names).

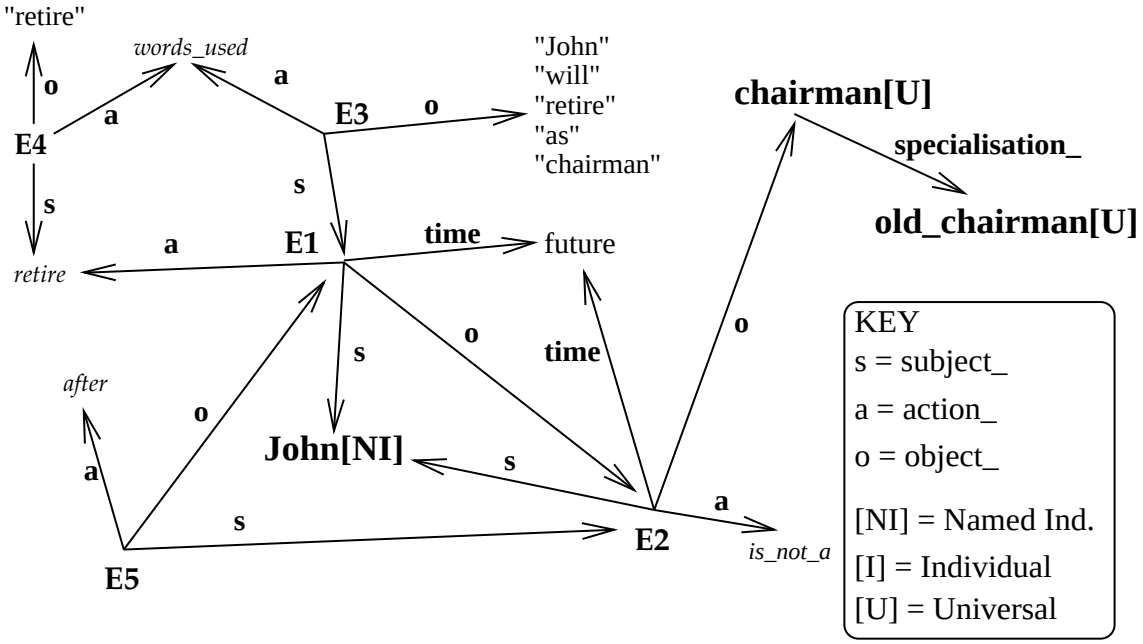


Figure 3.2: Example piece of Semantic Net: "John will retire as chairman". (See p. 19 for explanation.)

3.2.3 The Semantic Network

The net is a 96,000 node, directed 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 loosely characterises the meaning of the node (mainly used in debugging). Each link has an arc and a set of target nodes. Nodes correspond to concepts of entities or events. Links correspond to relationships between nodes. An arc is also a node, so 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.

Figure 3.2 shows a piece of semantic net. It is explained on p. 19, after we introduce the basic terminology. Detailed examples will be postponed until then.

Controls indicate basic information about a node, such as its type (eg event, entity, relation), its family type (see below), its lexical type (eg 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 entities (corresponding to constants) 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 depending on universal quantification. The family type is much used in the NE and TE tasks of MUC: possible values include *human*, *human organisation*, *temporal quantity*, and *location*.

The controls of new nodes are set during analysis by direct assignment (eg proper nouns identified as people will get corresponding controls), or by inference during semantics (eg a concept derived from an adjectified noun phrase will have the controls of the noun phrase, except where the adjective needs to override them). Controls could be represented using links, but for efficiency reasons this more compact form is used. (Some form of abstraction to hide this detail is needed.)

There are approximately 60 different arcs. The arcs `subject_`, `action_`, and `object_` are used to represent the basic components of an event. Note that these names are specific to LOLITA, and they do not represent grammatical notions of subject, object, etc. Events can have other arcs, such as those indicating temporal information, the status of the information (eg, known fact, hypothesis, &c), or arcs that indicate the source of the information. Most arcs also have inverses: eg the `subject_` arc has the inverse `subject_of_`, which allows determination of the

events in which a particular concept was a `subject_`. Such inverse arcs are essential in net-traversal algorithms, such as reasoning. Non-invertible arcs are those for which an inverse is not useful. For example, the status links are not invertible: quick access to all facts or all hypotheses in the system is rarely required, and storing such information is expensive.

Concepts may be connected with arcs such as `specialisation_` (and its inverse, `generalisation_`), or `instance_` (inverse `universal_`). Specialisation links a set to one possible subset, hence allowing specification of hierarchies of concepts. Other links between concepts include `synonym_` and `antonym_`. Figure 3.2 shows several arcs, as explained on p. 19.

More details about the formalism used in the net can be found in [Long and Garigliano, 1994]. The bulk of the net (65%) comes from WordNet, a database containing lexical and semantic information about word forms in English [Miller, 1990]. The net 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" &c. Use of inheritance in reasoning reduces duplication of information.
- **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: ie, 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. Eg, `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 components in an event. "Human owners own things" says that only humans can be a `subject_` in 'ownership' events. This is the main kind of pragmatic information in the system.
- **General Knowledge**: this is factual knowledge which may be useful in processing, for example "London is_a city", or "Roberto is_a professor". Often, such facts have been entered as natural language and the analysis results kept. Very little of this information has been manually built in the net.
- **Analysis Results**: LOLITA uses the net as a scratch-pad during analysis to store intermediate results. The final result of analysis is completely encoded in the net, and the intermediate results removed. For example, the content of a MUC-6 article would come in this class, when analysed. It could be kept in the net and further used – eg allowing the user to ask questions about the information.
- **Inferred information**: there are inference algorithms in the core to derive the implicit consequences of existing information, as explained below.

The goal of analysis is a disambiguated piece of net. Ambiguity is allowed in the intermediate results. If an ambiguous result remains at the end of analysis, one possibility is arbitrarily chosen, to avoid a combinatorial explosion of possible scenarios. Ambiguity is mainly in form of sense ambiguity and semantic ambiguity. The former is ambiguity between word senses, of which sometimes there may be many.

Explanation of Semantic Net Example

Figure 3.2 shows the piece of the semantic net which would be created for the event, "John will retire as chairman". Note that inverse links are not shown (eg `subject_of_` as the inverse of `subject_`), in order to keep the diagram simple.

'John' is represented as an object of rank *Named Individual*. He is the `subject_` of an event E1 whose `action_` is the concept of retiring, and whose `object_` is the state after he retires,

namely the event E2 that 'John' is not a chairman. Since John will reture, E1 and E2 are marked as 'future' events, and a further event E5 indicates that its `subject_` E2 is temporally after its `object_` E1, ie E1 preceeds E2.

The object `chairman[U]` represents the set of all possible chairmen, and the set of all possible old chairmen is represented by `old_chairman[U]`. Between the former and the latter is a `specialisation_` link, indicating that old chairmen form a subset of chairmen. Conversely, the latter is linked to the former with a `generalisation_` link, representing a superset, although this is not shown in the figure. If there was a particular chairman, eg denoted by `chairman1[I]`, it would be linked to `chairman[U]` through a `universal_` link.

The figure also shows some "textref events" (see section 3.2.5), which help to link surface text to semantic results. For example, E4 associates the string 'retire' with the concept determined during the analysis, and E3 does the same for the complete sentence. Note that these strings are particular strings from the input (ie, the extra information is not shown). The concept "words_used" is the `action_` in all textref events.

3.2.4 Semantic Net Reasoning Algorithms

- **Inheritance**: – this is the main technique, used to infer information about a node from its neighbours. A particular case is determining type information from the ancestors of a node.
- **Semantic Distance**: this is the distance (by summed arc costs) between two nodes. Each arc is assigned a weight. Basic event and hierarchy arcs have small weights, synonyms lower weights, and other arcs higher weights, which effectively removes them from consideration. One use of semantic distance is in template generation, to look for possibly-relevant concepts around a template base concept.
- **Analogy**: this is a form of plausible reasoning. Inference is by similarity of known properties between concepts. For example, A is a lecturer and he uses an office. B is also a lecturer. Does B use an office? On the basis of common information, the algorithm would agree, with moderate certainly. This algorithm is explained in detail in [Long and Garigliano, 1994].
- **Future**: several extensions are planned, including reasoning based on causality.

3.2.5 Referring back to the Original Text

Before MUC-6, LOLITA did not have a method of referring back to its input. Information about the language-dependent surface form was discarded as analysis constructed a language-independent logical representation. This did not cause problems in early work, but with hindsight, it is a weakness in the original design. Since the ability of referring back has many uses outside of the MUC-6 tasks, a more general mechanism was designed and added to the core.

New net nodes are allocated to components of the document (words, phrases, sentences, ...), which act as references into the document. Concept nodes and such 'textref' nodes are linked by an event with the internal action `words_used`. Two examples may be seen in figure 3.2: single words are attached to the main concepts of the sentence (only 'retire' is shown), and all of the textrefs in the sentence are attached to the node representing the whole event. The scheme has several uses:

- It enables applications to produce output which is highly related to the original text. Clearly, the SGML tasks are an example of this, since they require the original phrases and exact placement of annotations. Another possibility is the provision of hypertext-style links to the relevant parts of the original documents in information extraction or summarisation tasks.
- A more robust method of output: previous LOLITA applications have used the core's NL generator [Smith, 1996] to produce output. This generator relies heavily on the core analysis,

and although it performs well given a correct analysis, errors in the analysis can produce very strange output, and drastically reduce the perceived performance of the application.

- The textref system can also be used to provide convenient debugging information, since textrefs allow developers to relate internal structures produced by the system to the portions of the text from which they were derived.
- It allows the core to analyse input which talks about surface components of the input text.

3.2.6 Text Pre-processing

Core analysis of textual input starts from a LOLITA-specific SGML representation of the input. Individual applications must convert from their own formats (eg plain text, MUC-6 WSJ articles, LaTeX, HTML, ...) into this internal format. The MUC-6 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.

3.2.7 Morphology

Morphological analysis 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 (eg "I'll" expanded to "I will"), expand monetary and numeric expressions (eg "$10 million" to "10 million dollars"), and to translate certain surface-level idiomatic phrases (eg "in charge of") to internal concepts. Some splitting of hyphenated words is also done. The basic morphology function is then applied to all leaves.

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 analysis. 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 general 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.

3.2.8 Parsing

An example parse is given in figure 3.3, for the sentence of figure 3.2. There are four stages in parsing:

- A pre-parser which identifies and provides structure for monetary expressions. This stage is currently underused, and would provide a measure of robustness for the kind of expressions used in Named Entity, should parsing fail. It is implemented using a simple grammar of low ambiguity and a parser which attempts to find the largest non-overlapping sequences which match the grammar (working from left to right).
- Parsing of whole sentences using the Tomita algorithm [Tomita, 1986]. The main system grammar is large and highly ambiguous, so a powerful algorithm is required. Our grammar is written in a context-free style, using a simple feature system to parametrise pieces of grammar, and contains some rules for handling non-grammatical input. It is transformed into approximately 1900 rules of the type `A -> X` or `A -> X Y`, where `A` is a non-terminal, and `X`,`Y` can be terminals or non-terminals. The result of this stage is a "parse forest", a directed acyclic and-or graph which indicates all possible parses. Due to the complexity of the grammar, this forest is frequently very large, implying very many possible parses.

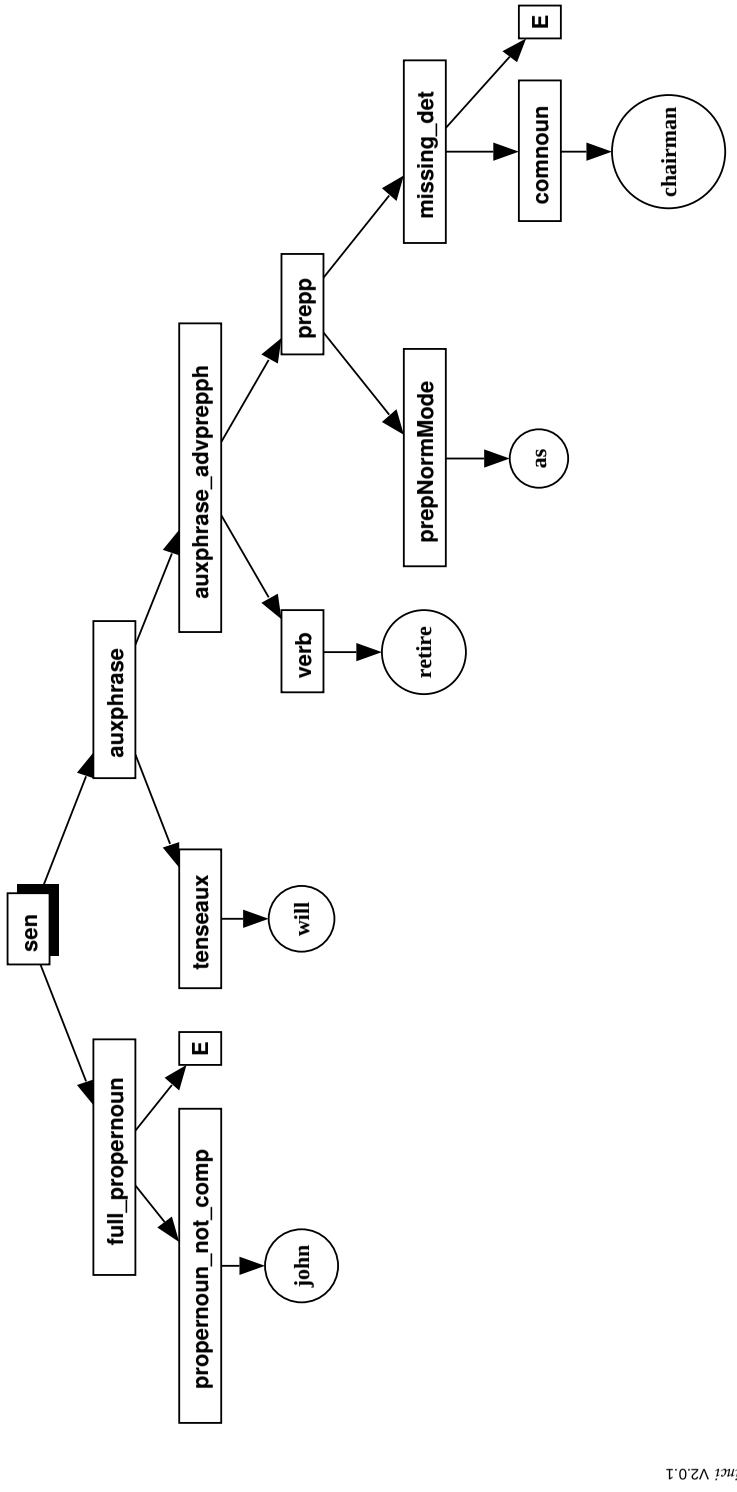


Figure 3.3: An example parse: "John will retire as chairman".

- 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.
- 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 deeper prepositional phrase attachments. It is possible for the subsequent analysis to reject suggested trees, and try the next best, but this option is not used in our MUC-6 system. Work to improve the handling of structural ambiguity is planned, possibly by passing a forest containing the lowest cost set to subsequent analysis.
- 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, ie "I was bitten by a dog" changed to "A dog bit me". Another class involves transformations such as "You are surprised" to "*SOMETHING* surprises you", which makes explicit the object doing the surprising.

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. We refer to expiry of the time limit as a 'timeout'. It is also possible for parses to fail if the sentence can't be analysed with the main grammar. If the parse fails, analysis is discontinued on that sentence – so no semantic result is produced. Notice that parsing failures can mean a serious loss of important information. We have no mechanism to avoid this at the moment.

3.2.9 Analysis of Meaning

This section describes how the parse tree is converted to a disambiguated piece of net. There are two stages, which we will call 'Semantic' and 'Pragmatic'.

The Semantic analysis is compositional in general: 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. 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 on the fly. In some cases, the semantic result of a subtree is labelled with the part it may play in the semantics of its parents. A good example of this is for verbs, where the parent will want to use the semantic result of the verb as the `action_` of a semantic event.

During this analysis, the textrefs for the text in a particular subtree are connected to concepts produced from the subtree, providing there is no ambiguity of reference. Such concepts may have textrefs already: this is the basis of coreference. The textref handling is completely invisible to the semantic rules.

The 'meaning' of most leaves (ie words) is the set of semantic nodes associated with the word at the Morphology stage. Tree branches are associated with rules for combining the semantics of their subtrees. For example, a phrase like "Alan Gottesman, an analyst with PaineWebber" is handled as follows. It is a proper noun phrase followed by a noun phrase describing that proper noun. Semantically, the subphrases give rise to nodes representing the named individual Gottesman and an arbitrary individual who is an analyst working with the named company, respectively. An event will be constructed in the net which has the former as the `subject_`, the latter as the `object_`, and the `action_` will be 'equality'. Thus, Gottesman has been identified as the particular analyst, and this fact can be later used as a candidate coreference link. The semantic result returned for the whole phrase will be the node for Gottesman.

The main task of the Pragmatic stage is further disambiguation and type checking. Lexical ambiguities and anaphora are resolved using a series of preference heuristics. Each heuristic eliminates any meanings which are not the 'preferred' ones. Given that the less favoured meanings

are rejected at once and no backtracking mechanism is used at present, the order of application of heuristics can have a big effect on the final interpretation. The order used has been developed by trial and error to get the desired meaning in the majority of cases in a small test set. In general, the cheaper heuristics are applied first, before using the more powerful but more expensive deep heuristics.

Examples of such heuristics include relevance to the topic(s) of the text (given to the system in advance, eg news reports), relevance to material in previous sentences, the amount of knowledge the system has about a given concept, or a concept's frequency of use. Some of the relevance tests are based on semantic distance, a measure of distance between nodes in the semantic network.

The process of "type checking" is applied when an event's `action_` is disambiguated. Prototype events encode information about a particular `action_`, which can be used in disambiguating the other components of the event, especially to rule out pragmatically implausible readings. Eg, for an event of 'owning', we can check that the `subject_` is a human entity and the `object_` is an animal or inanimate object.

The final part of pragmatic analysis is an attempt to establish connections between new events and the previously processed discourse. This includes creation of internal coreference links between entities and events.

3.2.10 Reference Resolution

The first stage of this is done "on the fly" in semantics. The context structure holds possible referents, ordered by recency of occurrence, and with semantic and feature information attached to aid disambiguation. Anaphoric expressions result in this context structure being examined for possible candidates which have appropriate feature and semantic information attached; if more than one candidate exists, then a new ('dummy') node is created to represent the alternatives and is linked to each of them. This new node is then returned as the meaning of the anaphor; later stages (eg Pragmatics) will attempt to disambiguate the reference, and will replace the dummy node with the chosen node.

A later stage of analysis examines the recently built pieces of net and attempts to connect those which are similar. This makes correspondences which were not picked up during the semantic analysis of individual sentences. A similar stage was added to help unify certain occurrences of proper names – cases such as 'FAA' and 'Federal Aviation Authority', and abbreviated forms such as 'PanAm' and 'Pan American'. In brief, the method looks for correspondences in the surface text attached to Named Individual nodes (ie, resulting from proper nouns). Furthermore, this process is used on titles: the grammar of article titles is quite different from that for normal text, so we avoided full analysis of titles and joined title textrefs to nodes when a surface match was found.

3.2.11 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 in the net, use of inference rules to derive implicit facts, and general output formatting. A fairly sophisticated facility existed pre-MUC-6 (as had a few template applications); for MUC-6, support for *hyper-templates* (templates that can refer to other templates) was added, plus – via the textref system – the ability to reproduce surface text.

A template contains a predefined set of slots with associated fill rules that direct the search for appropriate information in the net. There are currently four slot types, distinguished by how the slot output is produced:

- Concept Slot. This type of slot has associated with it a rule which produces a list of concept nodes with which the slot should be filled. Each concept node represents one slot fill and the generator, or the textref system, is used to express them in English.

- Textref Slot. Some concept nodes may have more than one related textref. In concept slots, some default rules are used to pick the most appropriate one, but for situations in which more control is required, the textref slot type allows its associated rule to define precisely the textref sequence to be used.
- String Slot. The slot fill rule chooses a string from a given list.
- Template Reference Slot. The output consists of a reference to another template, enabling *hyper-templates*.

Types of template can be distinguished:

- event-based templates – where one clearly identifiable event is the subject of the article. For example, a template regarding a *"takeover"* will include all the information (separated in different slots), referring to the takeover itself which represents the main event of the template.
- summary-templates – where the article does not contain a prominent event. The summary template is thus a collection of different kinds of information extracted from the source article. A summary template, for example, can consist of the slots: personal names, organisations, numeric expressions &c, found in the source article.
- hyper-templates. Hyper-templates are structures whose slots can refer to other templates, thus creating a graph of templates. This is similar to event-based templates, but the extra information appears in sub-templates instead of in the main template. Hyper-templates have been used for MUC-6 scenario templates.

Template calculation proceeds top-down. At any level there is a pool of nodes from which templates may be built. Templates are built if the conditions attached to them are satisfied for some node in the pool. Initially this pool is the complete analysis result for an article. When built, a template passes a smaller pool to the descriptions of child templates, and the process repeats. Typically, the main templates (eg SUCCESSION_EVENT) will pass to its children the set of its close neighbour nodes, such as all nodes within five arcs in the net (five was chosen by experiment). This allows recovery of information not directly connected to the main nodes, with the possibility of over-generating when non-relevant but close nodes are encountered. It is possible to write more complex rules which control the node pool more tightly. This process yields a collection of template pieces, some of which may not meet well-formedness constraints, such as a slot requiring just one child template link, or requiring certain slots to be filled in all template instances. These constraints are applied and the resulting template structure is converted to text.

3.2.12 Implementation of Named Entity

The algorithm works by examining the concepts created in the net following complete analysis. Much use is made of the control variables of a node (p. 3.2.3). The algorithm selects all new nodes which have a Named Individual rank control, which corresponds to all proper names and numerals (eg money, percentages). The control variable for family type is used to distinguish the type of concept which has been created, and subsequently the kind of markup to be added to the input text).

The Named Entity task is thus implemented as a predicate on a node's family type control, and is applied to all Named Individuals created during analysis. This suggests markable nodes, with an entity type. To each node will be attached zero or more textref sequences, which must be filtered on the basis of markability (eg, only proper nouns are markable), and illegal overlaps are removed. These textrefs are combined with the entity type and finally added to the SGML tree, which can be converted to plain text.

Although some experiments with substantial lists of company and place names were tried, these produced little improvement and were therefore not used in the formal evaluation. Some information on common human first names and surnames was already available in the net.

3.2.13 Implementation of Coreference

Like Named Entity, the Coref task begins with the set of all nodes created or modified during analysis. To some of these nodes will be attached a number of textref sequences. These are 'raw' Corefs, that is, they correspond to several pieces of text referring to the same concept. Then:

- The textref sequences are filtered to leave only validly markable ones, according to the MUC-6 Coref task definition of markability.
- In certain cases, nodes connected by 'is_a' (or, identity) links are marked as coreferential. The MUC-6 definition of what is coreferential differs from the notions implemented in LOLITA, so some non-trivial checking is needed.
- Remove the nodes which are partial heads: this prevents linking of 'cars' in the NP 'red cars and blue cars', but has to allow a link between 'sugar' in "I like sugar manufacturers because I like sugar". This rule was non-trivial to implement because the task definition was hard to interpret.
- Intersections between textref sequences of a single concept are removed. This is a robustness measure for when the core mistakenly produces duplicated textrefs.
- Concepts with less than two remaining textref sequences are discarded. Two references are needed to form a chain.
- The remaining concepts are converted to chains of markups, and then added to the SGML Tree.

3.2.14 Implementation of Template Elements

Using the general template facility, the ORGANIZATION template and the PERSON template are defined as *event-based* templates, since it is possible to find a clear underlying concept (person or organisation) from which to produce a template. Below is the definition of ORGANIZATION templates. Similar rules are used in the PERSON template.

```
> organisationTem
>  = addPostCond orgCondition                              $
>    addtextrefSlot "ORG_NAME"         nameFillRule        $
>    addtextrefSlot "ORG_ALIAS"        aliasFillRule       $
>    addtextrefSlot "ORG_DESCRIPTOR"   descriptorFillRule  $
>    addStringSlot  "ORG_TYPE"         orgType             $
>    addStringSlot  "ORG_LOCALE"       orgLocale           $
>    addStringSlot  "ORG_COUNTRY"      orgCountry          $
>    addDescriptorFun organisationDescriptor               $
>    addShowTemInstName showMUCTmplInstName                $
>    emptyTmpl   "ORGANIZATION" (TemFunctRule  isMUCOrganisation)
```

Reading from the bottom, the basic condition for a template is the predicate `isMUCOrganisation`: this checks for a node having an organisation type. The next line produces the name by appending the template node to the basic name – this explains why LOLITA produces long names. The next two are related: `orgLocale` checks the neighbours of the template node for location information, and `orgCountry` returns "United States" if some location is found, nothing otherwise. Returning that particular country is justified by typical MUC-6 texts.

`orgType` considers the generalisations of the template node (its supersets), and returns GOVERNMENT or COMPANY accordingly. Else, the test for the concept being mentioned in a phrase containing words like 'Administration' or 'Authority' returns GOVERNMENT. OTHER types are returned if the concept was mentioned in a proper noun phrase prefixed with 'the'. The default is COMPANY.

`nameFillRule` picks the longest textref phrase attached to the template node, filtering out phrases from the headline and those including possessives. `aliasFillRule` returns the shorter

attached phrases, plus those in the headline. `descriptorFillRule` examines the nodes around the template node which are essentially related by 'is_a' links. It returns noun phrases (not proper nouns), with filters to remove pronouns, phrases ending with ''s', and phrases used in the previous two name slots. Additionally, to cut over-generation, only phrases starting with a determiner are allowed - around 90% of the descriptors in a large set of keys did so. This is a trade of precision for recall.

3.2.15 Implementation of Scenario Templates

The scenario management template is defined using the *hyper-template* mechanism. The following is the definition of the SUCCESSION_EVENT template.

```
> successionEventTem
>  = addTemSlot "SUCCESSION_ORG" One organisationTem successionOrg  $
>    addAnyValueSlot "POST" positionTitle                           $
>    addTemSlot "IN_AND_OUT" OneOrMore inAndOutTem personInOrOut    $
>    addStringSlot "VACANCY_REASON" vacancyReason                   $
>    addShowTemInstName showMUCTmplInstName                         $
>    emptyTmpl "SUCCESSION_EVENT" (TemCondPP isSuccessionEvent)
```

A succession event is identified if an event has an action that can be generalised to a set of predefined "succession actions" (e.g. to dismiss, to fire &c) or can be itself identified as a succession event (e.g. appointment, promotion). Nodes within five links of the kind likely to connect concepts related to this succession event, are made available to the slots and sub-templates of this template. The wide pool is a compromise. Ideally, a much smaller pool would be required if LOLITA's analysis was highly accurate. But, in the MUC-6 articles, this is rarely the case – though relevant (and irrelevant) information is often connected in some way to neighbouring concepts. Hence it is a trade of recall (picking up more relevant concepts) against precision (losing by producing templates which belong to other events). The number of five arcs was determined by experimentation, though it may need review in light of more recent changes to LOLITA. There are no checks for relevance to the succession event; these could improve precision.

The SUCCESSION_ORG is filled by the organisation (as recognised by the organisation template, which was explained in the previous section) closest to the event. The POST is found by a search through all of the human concepts in the template-related nodes, examining the textref noun phrases attached to them, and filtering by conditions such as dropping posts involving 'director'. The post generated by the nearest node is used, with a default of "no title". For VACANCY_REASON, LOLITA only tests for DEPART_WORKFORCE and REASSIGNMENT, with a default of OTH_UNK. One of the first two is produced if a node is found in the related concepts which is an event with an appropriate action, eg someone retiring or someone stepping down, respectively.

The IN_AND_OUT fills, of which there can be 'OneOrMore', depend on producing an `inAndOutTem`. The current base condition is finding a person involved in the succession event. IO_PERSON is filled by a reference to the template for this person. NEW_STATUS is IN if the succession event has an 'incoming' action (eg promotion or appointment), OUT if it has an 'outgoing' action (eg fired or demoted), with IN as a default. ON_THE_JOB also examines the succession event, combining tense information and the NEW_STATUS to determine YES or NO. There is a possible bug in the code, as only past and future, not present, are considered: this may assume that newspaper articles should be analysed as all past events, unless a future event is explicitly indicated. If the event is one such as 'naming' or 'nominating', the status is UNCLEAR. The default is UNCLEAR. These conditions are clearly inadequate. No account is made of the roles different people play in the succession event, so all actors would get the same NEW_STATUS and ON_THE_JOB. This is confirmed in the next chapter. OTHER_ORG and REL_OTHER_ORG are unimplemented.

3.3 Work done in preparation for MUC-6

This section outlines the work done by the author as part of the LNLE's preparation for MUC-6. This is the 'implementation' part of the thesis. The most important work was done in Parsing, the Semantic Net, and in General Efficiency. The smaller tasks are discussed in the 'Miscellaneous' section. The 'Other Contributions' section lists work done by the author which were parts of larger efforts, with a final section outlining the work done by others in the group for MUC-6, for completeness.

In the context of a group, where sharing large pieces of work is the natural way of working, identifying the exact contribution of an individual is difficult. Design decisions are typically made by a small number of knowledgeable people, and implementation shared among a larger number. However, in the first four sections, the author undertook virtually all of the work described, both design and implementation, except in a few cases which are explicitly noted.

It should be noted that the author's work concentrated on the technical side of LOLITA, with practically no work on the 'rules' (eg grammar) or 'data' (eg net contents). Overall, the author estimates responsibility for between a quarter and a third of the work preparing LOLITA for MUC-6. (This does not take account of preparation work such as annotating articles, which was performed by others in the LNLE).

3.3.1 Parsing Work

The pre-MUC parser and grammar was implemented using Haskell and was based on the parsing combinator style of [Frost and Launchbury, 1989]. This was essentially a top-down backtracking parser. The grammar used a simple feature set, and over-generation was attacked using a simple penalty system: use of certain phrase rules or feature clashes induced a penalty which propagated upwards. Interpretations with low penalties were preferred. Some hand-produced optimisation had been added by applying some simple transformations to unify left-most tree segments, which resulted in fairly deterministic performance for some areas of the grammar. This grammar had been developed over several years by Prof. Garigliano.

The parser was slow, but the largest problem was maintaining the grammar in the presence of the hand-coded determinism. A project had attempted to automate the grammar transformations, aiming to produce an almost deterministic grammar from an arbitrary grammar [Ellis *et al.*, 1993]. A grammar parser and graph transformation engine (which executed a set of transformation rules) had been produced, but the main work was never completed.

The thesis author later demonstrated that the transformation task was effectively impracticable for a large-scale NLP grammar, and that contemporary work on ambiguous grammar parsing was superior. The transformation scheme was abandoned and the author prepared a prototype parser replacement based on a public domain implementation [Hopkins, 1993] in C of Tomita's algorithm [Tomita, 1986]. For this, a LOLITA grammar was first converted to plain CFG (Context Free Grammar) form and passed to the parser, along with the morphological analysis of a sentence. The parser returned a 'parse forest' (see figure 3.4) which was then decoded by a subset of LOLITA to produce a series of penalty ordered trees. This prototype was very successful, and was incorporated into LOLITA. Work then began to make the parsing process more efficient.

The grammar translation was implemented using the graph transformation engine. The grammar was actually a Haskell object (as in [Frost and Launchbury, 1989]), so had to be parsed to the equivalent graph, and some 'primitives' (such as optionals) unfolded into simpler constructs. The CFG was generated by a depth-first traversal of the graph, doing some renaming of labels used in different contexts to ensure uniqueness. Thus, the translation was not trivial. Several translation methods were tried to get better performance from the parser whilst working on MUC-6, often coupled with developments in other parts of the parsing system. For example, forest compression (see below) allowed a simpler representation of OR-nodes since after derivation, single-choice OR-nodes (of which there were many in all parses, a consequence of our style of grammar) were removed from the forest. Some modification was also required to avoid unfixable bugs in the C parser code. Observe that this non-trivial translation process allows easy specification of gram-

mar rules that apply to arbitrary fragments of tree. This is in contrast to the usual "local tree" style of definition (defining a rule in terms of its immediate descendents).

Conceptually, a parse forest is an AND-OR directed acyclic graph which represents all possible parses for the current sentence and grammar. Alternatively, it is the subset of the grammar which parses the current sentence exactly. AND-nodes indicate parse tree branches, and OR-nodes a local choice between possible derivations. LOLITA requires a single parse tree from this forest. The decoding technique involves depth-first traversing this forest, propagating features and applying penalties. The final result is a penalty-ordered list of parse trees. Decoding is problematic because of the size of the forest. A ten-word sentence can have several hundred legal parses in a grammar of good coverage. This can increase if the grammar attempts to handle 'errors', to assign structures to ungrammatical (with respect to everyday English) phrases.

The explosion of possibilities is faced by all people working on realistic NLP grammars, in some form or other. Figure 3.4 shows the forest for "I own a car". With the current grammar and lexicon, the forest has 250 nodes, which includes 76 choice nodes, leading to at least 13000 distinct interpretations (the decoding process ran out of heap at this point). However, the lowest cost set contains just the expected interpretation.

The following work was done to make parsing more efficient. Two main problems existed: a parse took unacceptably long to decode, or did not decode at all because it required too much memory.

- Improved the decoding process (designed and implemented by R. Morgan). The initial algorithm repeated the unpacking of a region of the forest each time it was visited. Storing past results is not straightforward in Haskell, due to lack of assignment. However, laziness allows the definition of structures which are self-referential without non-terminating or undefined recursion: the decoding algorithm was passed a table of intermediate results, each of which was the result of applying the algorithm to a portion of the forest. Evaluation occurred when a sub-forest was visited, and the result was available immediately at the next visit.

  This laziness of evaluation was managed easily in Haskell, with no additional book-keeping required. However, these methods are a trade-off between time and space: avoiding time-consuming recalculation by *storing* the decoded results requires memory. Many sentences still failed to parse.

  Auxiliary parts of the algorithm were carefully recoded by the author to use non-strictness more usefully. With help from a profiler (see section 3.3.3), further savings were made.

- Added C code to compress the parse forest before it is decoded, by removing all unnecessary nodes, such as all unary branches that could not appear in the final result and by collapsing OR-nodes with only one subtree. This reduced the size and memory requirements by a few factors, significantly reducing the number of failures for large sentences.

- Several bugs were found in the Tomita algorithm implementation. Hopkins was not surprised (email communication), and suggested that a re-implementation would be a good idea. One bug involved failure at the end of a sentence to reduce rules that had an epsilon on the right side. Since these epsilons were a consequence of the style of LOLITA grammar[4], the author modified the grammar translation to eliminate them by expanding the parent rules. This introduces an overhead in parsing by requiring extra rules in the raw CFG.

- Optimisation of the memory use and the grammar compiling routines of the C parser (assisted by C profiling tools).

- Adaption of the parser to use a dynamically supplied lexicon (ie, at run time). Formerly, the lexicon had to be supplied in advance as part of the grammar (at compile time). This had been a big limitation, as we could not transfer morphology and the LOLITA lexicon to C; even if we could, the memory requirements would have been severe.

[4]The grammar only allowed binary branches and leaves, so unary branches were *approximated* by having one subtree empty. The author was not party to this decision.

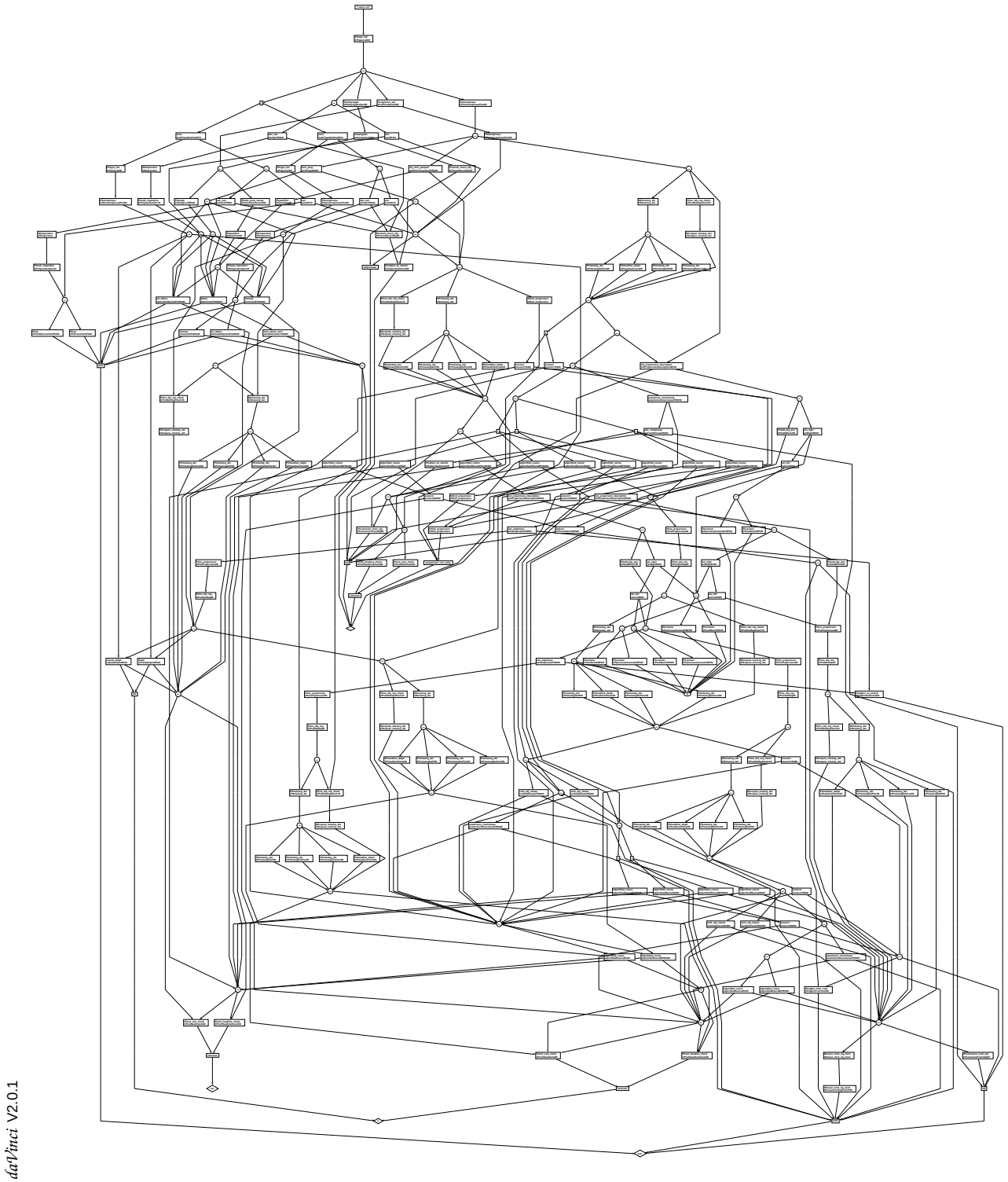


Figure 3.4: Parse forest (drawn using daVinci) of "I own a car". AND-nodes are boxes containing strings, OR-nodes are circles, WORDS are rhombi.

- Implemented grammar primitives that simplified penalty handling.
- Implemented tools to help check grammar consistency: names were sometimes repeated or omitted in the large grammar.
- Prepared a simple grammar development subset of LOLITA, allowing access to the detailed penalty information from decoding, and showing the differences between close derivations. It also allowed replacement of the 'top' symbol in parsing, which facilitated detailed testing of sub-grammars.
- Graphical displays of parse trees and parse forests were produced, using the daVinci graph display tool (see p. 37). Because of the size of the forests, daVinci was only able to display forests for very simple sentences (eg as in figure 3.4) with parse forests of several hundred nodes. This display was still useful as a visualisation of grammar over-generation, and a vast improvement on textual output. Parse tree visualisation is very useful.

The change of parser implementation resulted in a much faster system. Many further improvements in run time performance were subsequently obtained, but parsing of long and complex sentences remains a problem. However, only a few percent of sentences in MUC-6 articles fail to parse. We do not have backup mechanisms for when this happens, so important information can be lost.

3.3.2 Semantic Net Optimisation

Shortly before MUC-6 work began, the net was expanded from 30,000 to 96,000 nodes with information was derived from WordNet [Miller, 1990]. The increase in data size meant additional problems with sense ambiguity (since there were more meanings to disambiguate between), and algorithm cost (since search spaces were greatly increased). For example, common words like 'run' now had more than 10 senses. Furthermore, the storage and access to the data became a problem – both in time and space.

The data was previously stored in small chunks in ASCII files, and laziness was utilised to load data in to memory only when it was required. At this point, the text form was converted into an equivalent Haskell structure of lists and tuples. This technique, whilst conservative in resources when only a small part of the net was used, became a bottleneck when the net was heavily used. It could take a matter of minutes to load the whole net, and the data structure size was estimated at 40Mb minimum. Problems with unwanted laziness could increase this. Haskell arrays would have reduced some of the problems, but were not useful due to contemporary implementation problems with the compilers. There would still be overheads with access time and storage.

Since the compilers enabled it, it was decided to implement the main net storage in C. The net is stored as a C data structure, making heavy use of arrays and customised memory allocation routines to reduce the overhead of malloc-style allocation, with some data compression where possible[5]. Traversing node link structures is also a frequent operation in reasoning, so link storage was separated from the other data to avoid an indirection through a 'node' structure. The special cases of reflexive links were represented compactly. The English word dictionary was also coded in C, with hash-table lookup.

The total size of this representation is around 7Mb, a vast reduction from 40Mb. It may be completely loaded and saved in a few seconds. Access time has not been measured; clearly, it will have improved. Haskell-to-C calls provided a low-level interface to the C facilities, with operations like fetching the controls of a particular node, or direct access to the targets of a particular link on a given node. A simple referential transparency check was added for safety.

The net is the main data structure in LOLITA, and is used in a referentially transparent way. For example, parallel competing analyses can be processed by working on separate copies of the net. Referential transparency and the Haskell memory organisation ensures that in n copies of

[5]For example, there are only some 300 distinct combinations of controls in all of the 96,000 nodes.

the net, the memory demands are only of the order of the amount of change to the original net among the n copies, not of the order of n complete copies. This behaviour had to be preserved or improved on.

The solution is this: a Haskell-level tree structure acts as the store or buffer for processing-time changes to the particular copy of the net. Net operations consult this tree first. If the required information is not in this 'dynamic' portion of the net, then the C code is called to retrieve it from the 'static' part of the net. Else, the overriding value is returned *without* calling the C code. 'Updates' to the net produce a new Haskell structure from the previous one. Updates may be made permanent in the static net by calculating the differences between the dynamic and static portions, and then calling the appropriate C update functions. Some additional speed may be possible by periodically moving the dynamic data to the static net, eg between analysis of paragraphs.

Additional recoding of the net-using code was needed to take advantage of the new scheme. The following are examples. Some indicate that the previous code was lacking in abstraction as it depended on the details of the old representation, for example, checking target set membership with the basic list membership function.

- Link Selection – several algorithms involve net traversal along a small fixed set of links, ignoring the other links. An example is the inheritance process, which works mainly on the set relation links. Formerly, the code would have built in Haskell all of the links of a node and checked for the one required, and then discard the remainder – clearly doing useless work. The C code can go directly to the required link of a given node, returning just the targets in Haskell.

- Direct Access – certain algorithms can work directly on the static net with no or little loss of generality, so the dynamic net lookup could be avoided by a direct static net call. Semantic distance is one example, as most of its work is done on the static portion of the net. This searching can be done very quickly in C, with a small cost to the generality of the algorithm.

- Partial node changes – in normal use, only one or two links of a node change with frequency, hence storing a complete changed node is wasteful. It also introduces overheads in the net updating process. So, only the changed part of a node needs to be in the dynamic net. (not yet implemented)

- Target set membership – a frequent operation is checking if a certain node is in a link's target set. At present, this is done fully in Haskell, with an overhead of traversing a list and maybe of constructing it, if it is a static node. This can be replaced by a faster C call. (not yet implemented)

Further work was done with the `ghc` profiler (see section 3.3.3), which enabled targeting of net bottlenecks, particularly in the dynamic net tree. Apart from parsing, net operations are the main resource consumer. The timings of the inheritance and semantic distance algorithms were significantly improved. Since they were now cheaper on resources, they could be used more in disambiguation to produce better analyses, thus improving task performance. On a smaller scale, unnecessary node list building and discarding was reduced in heavily used code.

No formal testing of the improvements gained has been done. As in parsing, the improvements have been definitely noticeable to users, and a vast improvement over previous performance, so we did not need to measure this in detail. Furthermore, since the load on the net code varies with an article, and we do not know how uniform the use of the net is across articles, it it not clear how to objectively gauge 'success'. Neither is it clear how we may gauge it in an externally useful way. One possibility is to specify a quantitative test on performance: this is effectively a requirement on the system. But at this stage of research, such fixed requirements are hard to justify even if setting them is easy.

3.3.3 General Code Efficiency

Essentially, LOLITA was not previously used on large volumes of text, so its speed had never been a big priority in our work. But in development and testing for MUC which involved several developers with limited machine resources, and to run the whole final test set in a week, speed soon became very critical. Our debug-compile-test cycle was also too long, approximately 15 minutes for even very simple testing, and more thorough testing was frequently required, with a turnaround of hours. Additionally, several algorithms such as inference could improve results but were at the time prohibitively expensive. Improvements in efficiency would allow their wider use.

To improve LOLITA's performance, profiling tools were required, analogous to tools like `gprof` and `malloc`-monitoring for C. Unfortunately, profiling of functional languages is quite different from the C case. Non-strictness means that the 'work' of a function may be done incompletely and in pieces at different times: this complicates attribution of resource cost. Furthermore, the costs of evaluating a function's arguments (when required) should not be added to the cost of the function itself. Our then main compiler, `hbc`, had only heap profiling capabilities [Runciman and Wakeling, 1993]. This shows which parts of a program are holding on to large amounts of memory for some time-span, allowing programmers to check their expectations or intuitions about space usage. `ghc` offered more sophisticated facilities, such as time and space usage ([Sansom, 1994]). Additionally, a project in Durham [Jarvis, 1997] is modifying the `ghc` profiler to make it more flexible in use.

Glasgow helped with the initial conversion of code to `ghc`: they had been using a previous version of LOLITA to benchmark their compiler. The author updated their work for the contemporary version of LOLITA, including adding the `ghc` version of the Haskell-to-C interfaces, plus some work in adapting to the different library organisations. The basic `ghc`-compiled LOLITA was almost twice as fast as a basic `hbc`-compiled LOLITA. Optimised `ghc` was about twice as fast again, but since it traded memory for speed in several optimisations [Santos, 1995], it ran out of memory during parsing more often, too much to be reliable in MUC-6 work. This memory drain was across the whole program, so compiling the parser section without optimisation did not alleviate the problem.

The `ghc` profiler did not immediately work. The author discovered and then worked around two bugs in the profiler's handling of the Haskell-C interface, after some exploration with a source-level C debugger. Profiler use consisted of cycles of running an instrumented executable on several representative articles, studying the results, and then rewriting apparent inefficiencies or bottlenecks. The improvements in these areas were judged from the run in the next cycle. As noted above, heap profiles indicate the main memory 'hoarders'. This is only useful when there are serious problems. For finer work, time and allocation profiles are of more interest: `ghc` showed per function the call frequency, time spent evaluating it, and memory used in evaluating it. It also showed some global statistics, such as the extent of laziness and the amount of memory allocated in total. More specifically, the `ghc` model worked on the notion of a cost centre [Sansom, 1994] which could be placed at *arbitrary* points in the code, typically at the top of a function. This flexibility enabled detailed examination of code, or a rough view from key points in the program.

There is no established methodology for examining or for acting on functional language profiler results – such as the questions of when a function is a significant problem, or of what to do in such cases – so work was guided by intuition and experience. There is the additional complication of functional language compilation itself being a research project: the inefficiencies could be due to missed opportunities for simple optimisations, best cured at the compiler level rather than by complicating the source code. On the other hand, any improvement in speed with little work was to be welcomed.

The following intuitive strategy was used. A frequently-called piece of code needs to be as fast as possible, hence is a candidate for improvement. Less clear candidates are functions which take more resources than we believed they should, disproportionate to their importance in the program. Profiling was done in cycles. An improvement in one cycle would allow further candidates to rise to the top of the list of resource users. If the new candidates were amenable to improvement without much effort, then they would be rewritten. This process was repeated until the improvements gained became less important than other outstanding development work.

A two-pronged approach was used: complete profiling and partial profiling. Complete profiling involved attaching cost centres to all functions (done automatically by `ghc`). This allowed a view of the commonly used low-level functions. Partial profiling involved manually-placed cost centres at key points in LOLITA to gain a higher-level profile, for example at each of the major stages of analysis[6]. On occasion, an algorithm of particular interest was temporarily instrumented in the partial version. The two profiling executables were usually run in parallel, allowing two views of results on the same article and the same version of LOLITA.

Overall, the improvement work was guided by the author's several-year experience of programming in Haskell, by knowledge of how `ghc` would handle certain constructions and what forms it handles best (sometimes examining the compiler intermediate results), by discussion of snippets of code with the `ghc` developers, and by *experimentation*. In the absence of some reliable and easily applied metric on the 'quality' of a code transformation, experimentation of its effect in typical use is the best substitute. This is more so in non-strict functional programming, where transformations, though all provably correct, can have *varying* impacts on a program's time-space behaviour [Santos, 1995]. Thus, we were careful to make difficult decisions on the results from the LOLITA analysis of several MUC-6 articles. Quite often, improvements were easily visible in a single medium-sized article.

The profiling work resulted in the following successes and observations. Overall trends are reported first, followed by specific key examples.

- **Overall Memory Use**: the author obtained a big reduction in rate of memory use. Though most of this memory was allocated, used, and then very quickly reused because of advanced garbage collection techniques, there was still a slight overhead in the process.

  Most of the reduction was gained by small re-codings in the Semantic Net operations in cases where a list was produced from some value, tested, and then discarded. New functions which did the test on the original value were introduced. This is arguably an improvement in abstraction because there is less lower-level manipulation of values outside of the module exporting the type involved. It also supports the view that the flexibility available in Haskell can lead to a loss of abstraction in code.

  Another reduction came from the removal of 'splitting' in pattern matching. This is when a value is decomposed by pattern matching but then used in a further function call. Often, an expression to 'recompose' the value is used, at cost to time and memory. Modern Haskell allows pattern-matched values to have several names, so the whole value can be named and that name used in the next function call, avoiding the reconstruction. Most splitting in the code dates from before this language facility, when the alternative was a manual rewrite. This change therefore improved the code.

- **Avoiding non-strictness**: There were several cases of computations being done lazily, which were required soon after. Hence, it was better to avoid laziness and have the final result calculated *strictly*. Candidates were identified by unexpectedly high time or memory use, and judgement. Big culprits are rather noticeable, eg the dynamic net. Values which are used in a single-threaded way are also amenable. State values which are not widely shared, and which are often examined in predicates, are good examples. Big gains were obtained by making several of the basic global-state-manipulating functions more strict.

- **Program 'Exploration'**: LOLITA is a large program, and it is more or less impossible for a single person to read it thoroughly. Profiling helped to identify inadequacies and problems in old code, such as the net handling code. It gives a useful picture of what is happening in the program.

- **Unexpected recalculation**: LOLITA was originally written in a predecessor of Haskell which was more lazy than Haskell. Several pieces of code were tuned to working well under

[6] Though complete profiling included partial profiling, the partial results could not be easily deduced from the complete results. Unfortunately, this requires two separate executables, and two separate test runs. [Jarvis, 1997] aims at greater flexibility in profiling results by post-processing the results according to various criteria, thus requiring a single run of a single executable. This tool was not available during MUC-6 work.

the predecessor. The implications of this were not realised fully when the conversion was made, and until profiling was done, there was no indication of any detriment to performance. The basic symptom was unexpected recalculation of local values, which we expected to be evaluated once only. Though it was possible to get `ghc` to provide some of the laziness[7], the code was rewritten to be more Haskell-friendly.

- **In-lining code**: `ghc` allows small definitions to be expanded where they are used. This was done for several basic net operations and for some list utilities. This facility was not investigated much. For one thing, the required recompilation took too long to allow detailed comparisons.

- **Loop Catching**: Finding the cause of infinite loops is not trivial in Haskell. The imperative technique of inserting print statements has no direct equivalent. 'Traces' may be added to print out the value of expressions at various points, but their presence can affect the non-strictness of code and distort results, since to be printed, values may need further evaluation. If changes to the strictness of code propagate to the external interface of a module, extensive recompilation may be required by virtue of the current compilation technology. The problems can sometimes be avoided with a little care, but it is not straightforward.

  In contrast, the profiler can easily detect which part of the program is using increasing amounts of resources without terminating. Note that the detail of answer depends on the detail of the profiling. Very detailed profiles may indicate that time is being spent in certain list manipulating functions, which is hardly interesting. Sparser (partial) profiles will indicate the rough area of the program causing the loop. We found that our partial profiling executable gave sufficient information to focus quickly on the problems.

- **Tree Use**: In several places, trees replaced lists to provide faster lookup tables. The `ghc` library `FiniteMap`, providing a balanced tree was much used: it is used in the `ghc` compiler itself, so has been heavily optimised by the `ghc` developers to work well with `ghc`. The author implemented a tree indexed by `Int`s from a fixed continuous range, for which only an unpredictable subset would actually be used. This used laziness to provide $O(logn)$ lookup time in less than the space required for a conventional binary tree.

- **Dynamic Net Code**: Given the frequency of net operations, speeding up operations on the dynamic net was important. The author made the structure strict, and rewrote the access functions to be more efficient after a careful analysis of the code and design. Further improvements which allowed better handling by `ghc` were suggested by W. Partain (a `ghc` developer), including some which took advantage of specialised `sparc` machine instructions. As a result, the dynamic net code is very close in performance to the C equivalent.

  There is more work to be done on this structure. The design is biased towards operations during semantic interpretation building, where nodes are created and destroyed frequently. It is not certain that the design is amenable to the extensive use in the *higher* analysis phases, where lookups predominate. A switch to a simpler tree type for this higher work could show benefits.

- **Inheritance Reasoning and Semantic Distance**: Profiling showed several inefficiencies in these implementations. This information was a guide to the author during reimplementation of these modules, indicating frequently executed code and possible bottlenecks.

- **Parsing subsystem**: The memory problems are reported in the section above, as is the author's limited success in reducing parse failures with this approach. Under the idea that 'every little bit helps', the parse decoding module was profiled in detail, and changes that would be rejected in general work as being too small were attempted. Several useful changes were made in the feature handling. But overall, the order of parse complexity was too big to allow simple changes to have a useful effect.

[7] The so-called 'full laziness' transformation is a simple compiler optimisation, but it is not used widely in `ghc` because its effects, especially in context of other optimisations, are not well understood [Santos, 1995].

Another efficiency gain was in the execution of `ghc` itself: it transpired that intermediate files were being written to the disc drive containing the source files, so the author changed this to always use the much faster and local `/tmp` drive. In the presence of an unsatisfactory disc/network setup, this speeded up compilation significantly. A second improvement was to modify the object-code linking stage to produce the smallest possible executable by limiting disc drive load. This improved the debug-compile-test cycle by several minutes: the linking and writing to disc of a 12Mb executable is a time-consuming operation.

To summarise: as for the semantic net code, the improvements were clear and valuable, so we did not measure them in detail. Also as for the net code, no objective metrics suggest themselves, and fixed requirements would be hard to justify at this stage of research. Informally, vast improvements were gained in both run-time and compilation/linking time, most significantly at least a five-fold improvement in the former. We also found that LOLITA could now be run with very little heap (eg 4Mb) without much loss of speed (ie, loss through excessive garbage collection in the smaller heap) for articles of average size but containing few complex sentences. This demonstrates the reduction in standard memory requirements. There are still many possibilities for code improvement, and Haskell compiling technology is sure to improve – for example, the often-predicted parallelism which is a consequence of referential transparency.

However, we are still at a great disadvantage compared to our competitors in MUC-6 in terms of execution speed and development productivity. Some of this could be due to the additional complexity of analysis: LOLITA produces the most complete analysis of the systems in MUC-6[8]. But the significant factor is the current state of Haskell. None of the points below are permanent disadvantages, and improvements are possible with further research:

- The relatively slow speed of compilation, and the consequences of the Haskell module system, which can cause recompilation after strictness changes in interfaces. This is a consequence of Haskell being a developing standard, and of the compilers for it themselves being research projects.
- The quality of code produced – large, and not heavily optimised. For example, detailed strictness analysers could produce faster code when non-strictness is superfluous. `ghc` uses a simple analyser at present. Again, this is an open research area.
- Limited flexibility in profiling. The standard scheme in `ghc` requires trial and error to get good information from a large program. The scheme of [Jarvis, 1997] should help the profiling of large programs.
- Lack of specific NL tools.

The author wishes to thank the `ghc` group, especially W. Partain, for their fast and useful responses to our questions and problems during the work described.

3.3.4 Miscellaneous Coding

- **Implementing textrefs**: The basic design of these is explained in section 3.2.5. There were two main subtasks in their implementation: passing textrefs through the system, and the classification of textref sequences to grammatical type.

  The first task was complicated by LOLITA's original design of discarding surface linguistic detail as analysis proceeded. For example, information from function words could be transferred to features in parsing and the words dropped, or the word order changed in a 'Normalisation' phase. This normalisation involved mapping phrases to versions which were simpler for semantics to analyse, for example – passive to active, dative to prepositional. This both reduced the number of rule cases required in semantics, and enabled simplification of certain rules.

[8] This appears to be the case, after reading the system descriptions of the other entrants.

In semantics, the textrefs from a subtree get connected to new concepts produced from semantically analysing the subtree. The rearrangement of trees meant that textref sequences could not be produced as the union of textrefs in the subtree. Instead, the author designed and implemented the following scheme. Each subtree of the raw parse tree was assigned an unique identifier and the textref sequence from the subtree recorded. The tree was then passed through the final parsing stages and normalisation, where it was arbitrarily transformed. The node identifier was preserved by changing the abstractions used to define the normalisation stage, so no rule changes were needed. This change also enabled detailed debugging traces of normalisation. Afterwards, the surviving tree nodes were reunited with their textrefs and could be read off in semantics.

The classification facility was required to filter out un-markable textref sequences in NE and CO. The author implemented this by building a table of textref sequences mapped to basic phrasal type, which was consulted as markables were generated. With hindsight, a scheme which links textref sequences to the concept of their phrasal type in the net is superior. It avoids storing the table in the state, and keeps all the required information in the net. The basic phrasal type was determined from the original parse tree branch name for the subtree producing the textrefs. Producing the mapping was time-consuming and resulted in a few hard-to-detect bugs because the grammar names are not structured (ie, they are just strings), the grammar is not well-documented, and there is no strong theory behind the names.

- **Proper Name Database**: A system enabling mapping of arbitrary names to their type (Town, Person, Company &c) was implemented in C, with an appropriate Haskell interface. A 3.5Mb database was prepared from a public-domain company name database combined with the MUC-6 gazetteer. There were some complications to handle with matching rules, such as case distinctions. Since the core could infer the necessary information from other sources (eg context) in most cases, the database was deemed unnecessary.

- **Detailed Debugging Traces**: As work progressed, different developers needed non-trivial traces from their areas of the program from large test runs of LOLITA. Resources meant it was infeasible for each developer to run his own large test run. So, a method of separately collecting the various debugging information was required. The basic trace mechanism, described on p. 15, was not adequate. The author implemented a system by which the output of traces were written to specific separate files. For example, basic parsing information was written to `X.parse`, the daVinci graph version of the trees to `X.graphs`, and Coreference details to `X.coref`.

- **Graph Displays using daVinci**: Graphs and trees are much used in LOLITA, so good ways of viewing them are highly useful and aid development. Previously, the presentation had been text only. Of the several public domain graph display tools available, daVinci [Fröhlich and Werner, 1997], the result of ongoing Ph.D. research in graph layout algorithms, provided the easiest to use and most reliable facilities – such as basic drawing of nodes and arcs in several styles, a layout algorithm with customisable parameters, some manual adjustment of the layout, and production of diagrams[9].

  Additionally, daVinci can be used remotely as the front-end for a graph-producing application. The author made use of this facility to produce a simple graphical front-end for LOLITA, and a net browsing tool. The latter has been indispensable in analysing LOLITA output for chapter 4, especially when large pieces of semantic net needed to be analysed (see p. 64).

  Haskell types were written for the daVinci application interface, and code to convert these to the exact form required by daVinci. The consequence of this is that graphs can be built and manipulated easily as Haskell values before being sent to daVinci. Code to convert normal parse trees and parse forests to graphs allowed the display of these. Unfortunately, the sheer size of parse forests mean that only forests of short sentences can be displayed (see figure 3.4). The parse-tree display was very useful in checking bracketing in large parses.

[9] This has been used several times in the thesis.

daVinci was also used to examine the trees output as debugging traces in test runs. It was coupled with a simple `perl` script which used the application interface to augment daVinci's functionality: this helped viewing of large trees by displaying only small sections at a time and allowing unfolding of the hidden sections by a double mouse click, and helped movement around trees scrolling clicked nodes to the window's centre.

LOLITA's use of daVinci was featured at a graph-drawing conference as a real application of daVinci.

- **Pre-parser**: Late in development, it was noticed that some of the named entities were being parsed as heavily ambiguous. Their descriptions, such as monetary expressions, were part of the main grammar – but parts of these expressions were being accepted by other areas of the grammar and in many cases, the incorrect analysis was used.

  Since most of the named entities can be parsed by a technique simpler than the Tomita algorithm used in the main parser, we tried to pick out these before the main parsing stage by using a top-down, backtracking parser (written in Haskell) which had been intended for use with the automatically transformed grammar (p. 28). The author undertook the necessary additional implementation and modification to the main parser. The initial time performance of this pre-parser was poor, so time was spent on hand-optimising it; at the same time, a parallel effort on improving the main grammar was underway.

  In last-minute tests, the pre-parser was not recognising significantly more named entities than the main grammar and it was slower, so was not used in the final system. (It is now used for most named entities, and is a significant part of analysis).

3.3.5 Other Contributions

- Scripts: Several scripts were required for running LOLITA on the required files and to analyse or summarise the results. Particular cases were the highlighting of important error messages, and reporting of important statistics. Another facility was running multiple LOLITAs on a multi-processor machine, such that the next article is analysed when a processor becomes free. This is important for throughput of testing. The author wrote several of the initial versions of these, before they were extended for general use by other people.

- Coreference implementation: The author extended the original code to use the textref classification facilities which he previously implemented. One of the inference algorithms which calculated task-specific is_a relations was rewritten to be more efficient – the original was repeating expensive net operations.

- Marking Coreference Chains: The original algorithm produced a target markup to which all other markups in the chain pointed. It was discovered that the coreference scorer was (unfairly) sensitive to the style of chaining, and preferred the "flat chain" style where A points to B, B to C &c. Occasional problems with textrefs meant adding a markup could fail. This affected the flat chaining of markups, since a chain could be broken. The author implemented a robust version of the chain-adding.

- Main commands for running the MUC-6 tasks: The author implemented the scheme by which arbitrary MUC-6 task results could be produced from a completed analysis. It used partial application to reduce all MUC-6 tasks to functions of identical type, despite their different output production methods.

- Improving the Pragmatic checking stage: This code was designed to discriminate among a set of possible concepts by applying successively more restrictive (and more expensive) tests until one concept was left. The early tests performed simple family checks (eg animate vs man-made). The later ones involved inference or semantic distance. The author re-implemented the algorithm, introducing an abstraction for pragmatic rules which helped avoid needless recalculation. A good increase in performance was gained.

- Last but not least: Throughout the MUC-6 work, the author undertook many smaller debugging jobs in many areas of LOLITA, and helped other people in their work. The analysis of a few small texts was checked in detail, uncovering several bugs and omissions in the rules.

3.3.6 Brief summary of other work done

This section briefly records the work done by others in the group, to give an idea of the complete effort.

- Writing an SGML parser, and modifying the pre-semantic stages to work on SGML trees.
- Extension of the template mechanism to allow hyper-templates.
- Implementation of textrefs.
- Alter the representation and manipulation of word sense ambiguity in the net.
- Incorporation of data from WordNet [Miller, 1990].
- Interfacing to Brill's tagger [Brill, 1995], and mapping from its tag-set to LOLITA's syntactic categories. This was not used in the final evaluation.
- General debugging.
- The original task implementations themselves.
- Annotation of texts (some of the dry run and training data annotation was done by participating sites).

3.4 Conclusions

3.4.1 Suitability of Haskell

Functional languages, in particular Haskell, are an uncommon choice for implementing an NL system. We shall consider their suitability with respect to the MUC work. The key point for us is that the edit-compile-test was long, even with the improvements just described. The machine resource requirements meant that the problem was worse with several people working simultaneously. Secondly, most of the frequent code changes were changes to rules, and carried out by experienced programmers, so the abstraction and type support was not essential. Such disadvantages negated the benefits of using the language.

This is not to say that functional languages is a bad choice for NL work. The case is more that it did not prove helpful in the circumstances, and in the way in which the language was used. It is apparent from the reports of other participants [DARPA, 1995] that a prototyping approach was common, and that the edit-compile-test cycle time was a key factor.

A possible solution to our immediate problems is a change in the implementation of LOLITA: the rules which need to be developed in such a way can be 'externalised', or made more independent of the Haskell code, and compiled in at run time. This happens in a weak sense for the grammar. There would be a trade-off with run-time speed, but such rules could be made concrete when the immediate development is finished.

3.4.2 Conversion of LOLITA for MUC-6

We consider the process of adapting LOLITA to the MUC-6 tasks. The ease of *implementation* of new tasks is important evidence for the claim about the worth of LOLITA, as well as how good the resulting performance is. There are two issues: changes required in the core system, and the

implementation of the tasks by using the core. The tasks necessitated some changes in the core. We note the following:

- Several big design changes were needed: the textref system and use of SGML trees as the main text representation. The former was more problematic, so we discuss that. The work took time because preservation of text was not considered in the original design, which discarded or rearranged surface information as analysis progressed. Some of this was countered by implementing a form of record-keeping underneath the abstractions, but a lack of abstraction in other portions meant that this information could be duplicated or omitted, and the cause was hard to find. Debugging became a time-consuming case-by-case testing of suspect areas. We did not have the time to rewrite the code to a higher standard. These are problems with design and implementation, so they do not affect the idea of LOLITA per se, though they did hinder work.

- There were many problems with integration of rules, which is due to the derivational style of analysis: rules could mask others, or conflict. A particular problem was the output of new rules producing a state which was not matched by existing later rules in the sequence. Such bugs require careful analysis to detect. Some support can be gained by changes in implementation, but problems may still remain whilst the basic architecture is derivational.

- The grammar is a particular case: the difficulties with the maintenance of large NL grammars are well known, so it does appear a weak idea to modify an existing grammar to parse constructs from a new domain (WSJ news articles) within a short development time, especially when development support is poor. That many changes were found necessary does indicate that the original grammar was not general. This weakens LOLITA's claim of generality.

- Analysis of errors showed a need for rules which we found could not be easily expressed in the existing frameworks. This meant a decision on whether to adapt the framework to allow a principled solution, to ignore or try to work around the case, or to try some heuristic fix.

- The size and complexity of LOLITA meant that time was spent in hunting for obscure bugs. There was little support in the code for detecting anomalous conditions or for tracing analysis.

- Task implementation was relatively straightforward. Some problems were encountered when the task's notion of a phenomena differed from the notion implemented in the core (such as the coreference relation).

3.4.3 Success of the author's work

The author's work, mainly on the technical side of LOLITA, has been very successful. Significant decreases in time and space requirements have been obtained, which means that testing is less of a bottleneck in the development process. Also, the final evaluation was run comfortably within the allotted week.

Informally, the kind of overall improvement obtained is estimated as at least a ten-fold improvement in run-time on a medium-sized article (3k) relative to a pre-MUC-6 `hbc`-compiled executable, plus gaining a more maintainable grammar and the possibility of working with a much reduced Haskell heap size. This is despite a three-fold increase in the net size.

Chapter 4

LOLITA Performance on MUC-6: A Specimen Article

4.1 Introduction

This chapter considers in detail the results on a single article. For all four tasks, and for all features in the answers for those tasks, the reasons why LOLITA gained points or lost points are discussed. The current performance of LOLITA is also discussed, as the result of approximately one person-year's worth of work by various people on the major problems raised by the MUC-6 results. Finally, conclusions are made on each section and then overall. The latter are brief, but will be extended in the conclusions of the next chapter, after a consideration of the overall scores. This chapter has several aims:

- To illustrate how LOLITA calculated its answers in a style which is more accessible to people interested in MUC-6 than in a discussion of LOLITA internals. It is obviously impossible to include full details of the computation due to the complexity of LOLITA; the author hopes that the detail provided gives some insight into how answers were reached.
- It provides a bridge between the technical detail of the previous chapter, and the score tables of the next chapter. It introduces some of the terminology and information about the MUC-6 tasks which is useful in understanding the next chapter. However, this chapter and the next assumes some familiarity with the MUC-6, such as detail about recall, precision, f-measures, &c. The proceedings [DARPA, 1995] should be consulted for more information.
- It also helps to illustrate the demands on processing, or the difficulty, of the MUC-6 tasks. We do not know of any such analysis in the literature. In particular, we outline what kind of processing is needed to obtain answers.
- We compare LOLITA *qualitatively* to other MUC-6 participants, using the template comparison tool described in Appendix B. This comparison is done anonymously: we are only concerned with LOLITA's results[1]. We shall be looking for places where LOLITA is in the correct minority (as evidence of good performance), where it is in the incorrect minority (as cases where easy marks are being lost), and will be interested generally in the errors made by all systems. For example, over-generation can occur when usually reliable, but simple, rules fail to make deep distinctions.

[1] The other system results, map histories, and score files were obtained from the MUC-6 ftp site. All entries to a task are considered, including multiple entries from the same site. Some are omitted in NE because of their low scores.

4.1.1 Choosing an article

The article was chosen by examining the distribution of scores and article sizes. The intention was to find a representative article, in both size and score. The first limitation was to the Scenario-template-producing articles of the thirty used in the CO-NE test. Then the scores for those fifteen articles were compared against the recall and precision scores for the other tasks. The aim was to pick an article inside the inter-quartile range for all tasks, though the NE task was considered less important than the others. Since the recall and precision scores do not reflect the number of scorable features in an article, article size and the number of possible Coref links were also considered. The choice was made using graphs of recall against precision, figures 5.2 to 5.5 in the next chapter.

The choice was not straightforward since there seemed little correlation between scores in different tasks[2]. No article was within the recall and precision inter-quartile ranges (IQR) for all tasks. The chosen article, number 9306220057 and listed in Appendix A, is inside or very close to either the precision IQR or recall IQR for all tasks except ST, and has a near-median word and Coref link count. The ST precision is low, with high recall. This article is marked `ch` in the graphs of chapter 5, and is 49th in size order, out of 100. All sentences produced a parse in this article, which means all sentences were analysed (grammar incompleteness or run-time constraints – "time outs" – can mean that a sentence is rejected). We did not wish to re-use the walk-through article (number 9402240133, marked `wt` in the graphs, 91st in size order). Scores were poor for this article, especially in CO. More importantly, the article was considered too large for the detailed analysis done here.

4.1.2 Preliminary Comments

Some points are raised here, to avoid repetition in the main discussion:

- In all tasks, the correctness of the keys is assumed, though we have noticed errors in the ST and TE keys (see section 6.2.3).
- In the discussion, we consider LOLITA performance first, then make comments on how the other competitors, or 'systems', performed on that portion of the key. We refer to items within the key as 'features' of the key.
- In the multi-system comparisons, all entries are considered – which includes several entries from the same site which may have similar results. Counts of errors do not distinguish several sites being incorrect from one site making an error in its basic system. Therefore, care must be taken in interpretation. Informally, it appears that the multiple entries were from strong sites, so this is not a large problem. For convenience, the terms 'entry' and 'system' are used interchangeably.
- In general, LOLITA's analysis is monotonic in the sense that later information does not cause a widespread revision of earlier decisions. Early analysis results are often in an under-specified form (eg, a set of possibilities). If further information does not cancel out some of the options, one possibility is chosen from those remaining at random, usually after analysing each sentence. Not doing so can result in a combinatorial explosion of possibilities.
- LOLITA is not well-instrumented to determine easily the exact cause of a feature in the analysis results or the output derived from it. One reason for this is the complexity of the system. Another is the difficulty with which such debugging aids can be added to a functional program, both in terms of use and of effect on development time (ie, recompilation). These issues have been discussed in section 3.1. The consequence is that several of the explanations of how certain answers were produced are sometimes vague. To produce better explanations would require much testing and ad hoc instrumentation. The author

[2] This has not been tested statistically. Correlations between *two* sets of data are considered in the next chapter, but these tests are not transitive. No analysis of simultaneously more than two data sets was attempted.

justifies this vagueness by noting that these details are not central to the thesis, but are supplied only to give an idea of the workings of LOLITA.

4.2 Named Entity

Below is a (condensed) summary of the NE performance, generated by the template comparison tool. NE is normally scored by first converting the SGML to a template form, and then scoring as for the usual template tasks. The C version of the NE scorer was modified to output the NE templates as they are being built, and a small `perl` script formats this information to resemble conventional MUC templates. The scorer also produces a map file. This map is then fed with the resulting templates to the comparison tool. The Named Entities of each type are shown in occurrence-in-key order.

```
## TYPE:                ORGANIZATION                ENAMEX      50.00
                XX  L= PERSON
   TEXT:                "Johnson & Johnson"
-------------------------------------------------------------------------
   TYPE:                ORGANIZATION                ENAMEX      100.00
   TEXT:                "Genetic Therapy"
-------------------------------------------------------------------------
   TYPE:                DATE                        TIMEX       100.00
   TEXT:                06-22-93
-------------------------------------------------------------------------
   TYPE:                LOCATION                    ENAMEX      100.00
   TEXT:                GAITHERSBURG
-------------------------------------------------------------------------
   TYPE:                LOCATION                    ENAMEX      100.00
## TEXT:                Md.
                XX  L= Md
-------------------------------------------------------------------------
   TYPE:                PERSON                      ENAMEX      100.00
   TEXT:                "Michael D. Casey"
-------------------------------------------------------------------------
## TYPE:                ORGANIZATION                ENAMEX      50.00
                XX  L= PERSON
   TEXT:                "Johnson & Johnson"
-------------------------------------------------------------------------
   TYPE:                ORGANIZATION                ENAMEX      100.00
## TEXT:                "Genetic Therapy Inc."
                XX  L= "Genetic Therapy Inc"
-------------------------------------------------------------------------
   TYPE:                PERSON                      ENAMEX      100.00
   TEXT:                Casey
-------------------------------------------------------------------------
## TYPE:                ORGANIZATION                ENAMEX      50.00
                XX  L= PERSON
   TEXT:                J&J
-------------------------------------------------------------------------
   TYPE:                ORGANIZATION                ENAMEX      100.00
   TEXT:                "McNeil Pharmaceutical"
-------------------------------------------------------------------------
## TYPE:                ORGANIZATION                ENAMEX      50.00
                XX  L= PERSON
   TEXT:                J&J
-------------------------------------------------------------------------
   TYPE:                ORGANIZATION                ENAMEX      100.00
## TEXT:                "Ortho Pharmaceutical Corp."
                XX  L= "Ortho Pharmaceutical Corp"
-------------------------------------------------------------------------
   TYPE:                PERSON                      ENAMEX      100.00
   TEXT:                Casey
-------------------------------------------------------------------------
   TYPE:                PERSON                      ENAMEX      100.00
   TEXT:                "M. James Barrett"
-------------------------------------------------------------------------
```

```
   TYPE:              ORGANIZATION              ENAMEX      100.00
   TEXT:              "Genetic Therapy"
-------------------------------------------------------------------
   TYPE:              PERSON                    ENAMEX      100.00
   TEXT:              Barrett
-------------------------------------------------------------------
   TYPE:              PERSON                    ENAMEX      100.00
   TEXT:              Casey
-------------------------------------------------------------------
   TYPE:              PERSON                    ENAMEX      100.00
   TEXT:              Casey
-------------------------------------------------------------------
## TYPE:              ORGANIZATION              ENAMEX      50.00
              XX  L= PERSON
   TEXT:              J&J
-------------------------------------------------------------------
   TYPE:              ORGANIZATION              ENAMEX      100.00
   TEXT:              "Genetic Therapy"
-------------------------------------------------------------------
   TYPE:              PERSON                    ENAMEX      {none}
   TEXT:              "John T.W. Hawkins"
-------------------------------------------------------------------
   TYPE:              ORGANIZATION              ENAMEX      100.00
   TEXT:              "Genetic Therapy"
-------------------------------------------------------------------
```

Recall (85%) is better than the upper quartile, and precision (89%) in the third quartile (ie, between median and upper quartile), so this is better than NE performance on all thirty articles (remember that NE was seen as the least significant task when making the choice).

For comparison against other systems, unlike the other tasks, there are entries in NE which score much less than LOLITA. These miss many markups, so their results in the comparison is essentially noise: it is no surprise when they miss a certain markup. Since our main interest in comparison is with systems *better* than LOLITA, the three lowest scoring systems on this article (with recall 0, 17 and 26) have been omitted. Except for one system with recall 61, the remaining sixteen score above 80.

Entities with errors will be considered first: there are six, of three types:

- Incorrectly assigning a `PERSON` type to both occurrences of the `ORGANIZATION` "Johnson & Johnson". The net contains several common surnames, which are typed as human. Thus, the string is initially interpreted as a pair of two people. However, the phrase "X of Y" triggers a link of the concept for organisation to Y during the semantic phase (here, X is "manager"). This information is not merged correctly with the person information, hence the PERSON type.

  Three systems miss both occurrences, one systems is wrong with both names (producing 'Johnson'), two other systems make this mistake in the title occurrence, and two systems (including LOLITA) are wrong on both types (the other system said LOCATION). All other systems are correct.

- Incorrectly assigning a `PERSON` type to all three occurrences of the "J&J" `ORGANIZATION`. A heuristic has been implemented to match phrases such as "Alpha & Beta" to 'new' symbols in the form "A&B". That is, on the basis of initial letters in a proper noun phrase. This was added to the core in response to the frequency of that abbreviation style in the WSJ. Hence it produces a link between concepts of "Johnson & Johnson" and "J&J", and so, all errors of the full expression are inherited by the abbreviation.

  One system misses all occurrences. One system misses the type in two and is wrong in the other. Four systems are misled in the first occurrence, marking the string "J&J's McNeil". LOLITA is the only system with the type wrong in all cases. The remainder get 100%.

- Omitting the `PERSON` "John T.W. Hawkins". The string is correctly parsed as a proper name, but the initials are wrongly interpreted – a flaw in the name handling. 'T' gets resolved to the Terabyte unit, and 'W' to the metal tungsten. This causes a clash with

the information about John being the name of a human or animal, resulting in assigning the node for Hawkins a type less specific than human. Thus, Hawkins is never detected as possibly human by the NE-specific code. One other system missed this markup, with five more getting the name wrong (four missing the surname and one missing the forename and first initial) and one omitting the type.

The remaining entities were correct for LOLITA. Below, they are grouped by similarity:

- `DATE 06/22/93`: this is easily recovered by virtue of being inside the `DD` tag, and by its syntax. Only one system missed this.
- `LOCATION GAITHERSBURG`: easily recoverable from the `DATELINE`, as is the `LOCATION Md`. One entry misses the city and another decides it is a person. The state abbreviation is omitted by the second system, and the first system marks the whole DATELINE phrase. Note that this system was awarded a match for correct type but incorrect text.
- `ORGANIZATION "Genetic Therapy (Inc)"`: this expression, with its capitalised words, is easily parsed as a name. The first occurrence creates the concept of something with that name. The presence of "Inc" leaves it with the correct type. Subsequent mentions link to this concept, hence propagating the type. The occurrence in the title is linked after analysis on the basis of word matching to the text attached to a concept. No system produces the wrong type for these markups, and most systems get all five markups. One system misses three of these, getting the 'Inc' and the last occurrence.
- `ORGANIZATION "McNeil Pharmaceutical"`: capital letters indicate a proper noun. After semantics, the resulting concept has not been assigned a type, so after simple checks on the node's neighbours, a default type `ORGANIZATION` is assumed. Eight of seventeen systems miss this markup, and the remainder get full marks.
- `ORGANIZATION "Ortho Pharmaceutical Corp."`: it is recognised as a name on the evidence of capitalisation. The type is assigned through the suffix 'Corp'. All systems get this markup.
- `PERSON "M. James Barrett", Barrett`: obviously names; the type is suggested by the known forename and surname. The isolated surname occurs afterwards, and is connected to the earlier-built concept. All systems are correct here.
- `PERSON "Michael D. Casey", Casey`: the full version appears first, allowing creation of a concept with human type, due to the name and surname. The remaining occurrences of 'Casey' get linked to this matching entity in the list of possible referents. Again, all systems are correct.

LOLITA did not over-generate. Other interesting over-generations are that three systems over-generated with a person 'Johnson'; these appear to be the unmatched halves of a pair of persons produced from "Johnson & Johnson". Three systems suggest a person 'Makes' or 'Makes Move' from the title.

4.3 Coreference

Coreference results are harder to analyse, the structures and linkages produced being less straightforward than templates. However, coref SGML is convertible to templates. Indeed, this step is part of the scoring process, as for NE. The `emacs` version of the scorer is easily modified to save the resulting templates before the usual mapping begins, and the map history itself can also be saved. This means the template comparison tool may be used. The coref mode for this tool (Appendix B.3) adds a new slot `KEY_CHAIN` to the key and a slot `OWN_CHAIN` to the response, marking which of the key or response equivalence classes the template belongs to. The output shows each key chain in turn, followed by the unmapped templates grouped by the TEXT slot.

Some manual rearrangement of template order has been done to ease comprehension, such as placing unmapped response templates near the key chains they should have mapped to.

However, the map history does not contain f-scores: though the recall and precision scores could be used in the normal formula for f-scores, it is not yet certain if this number is meaningful ([Chinchor, 1995b] p39). The author is currently experimenting with ways of presenting information produced during scoring: the scheme used here has been found useful, especially in assessing *changes* in performance.

We assume the scoring method described in [Vilain *et al.*, 1995]. Recall scores are a property of equivalence classes in the key which indicate how many of the links in this class were identified in the response. Precision scores are a property of equivalence classes in the response which indicate how many of the links in this class were present in the key. If a template is mapped, then we give it the recall score of its key class and the precision score of its response class. Still, we can't easily tell if a particular markup scored in that key chain, so extra information about the *intersection* of the particular key and response chains is shown. This is a finer grain of detail.

For example, with reference to the discussion below, key markup `COREF-33` (on p 52) maps to response markup `COREF-1000003` with 'score' `sc R:2 P:2 => r:8/16 p:2/3`, which is interpreted as follows. For the (key) chain which contains `COREF-33`, there are sixteen possible coreference links, of which LOLITA has got eight correct. These links are shared across several chains in the response, and this particular intersection of key and response accounts for two points of the eight for recall. Similarly for precision, the response chain contains three coreference links, of which only two are correct; the intersection of interest contains both of these correct links. Response markup `COREF-1000001` (p. 49) is mapped to a markup in chain `K_0`, but no other template from its response chain (`L_4`) is mapped into `K_0`, thus it does not contribute to `K_0` recall but it does reduce precision for `L_4` – part of which does score in `K_6`. Response chain `L_9` (p. 51) does not contribute to scores anywhere (see `K_5`). Referring to the diagrams overleaf (and see below) may help to visualise the situation.

The recall score is immediately useful for a display sorted by key class (as below). The precision score is less relevant since it is not constrained by the key class.

Another method is to graphically display the relationship between key chains and response chains. Figures 4.1 and 4.2 show the pattern for the specimen article. The chains are grouped by interdependency – ie, response chain `L_0` maps to templates in both of the key chains `K_6` and `K_8`. The diagrams were produced using daVinci ([Fröhlich and Werner, 1997] and see p. 37). It may be useful to refer to these diagrams in the following discussion, to get an overall picture of the cross-linking.

The output below has been scored with the optional markups present, leading to scores which are slightly lower than the official scores – however, we do not produce any of the optional markups, hence the only difference is in the number of possible coref links (the denominator of recall). The performance of the other six coref systems is discussed at the end of each key chain section. The tool output is too long for inclusion, but may be obtained from the author. Note that the `emacs` scorer outputs templates in upper case, hence the upper case in the slot values.

The article is near upper quartile for precision (0.56) and above upper quartile for recall (0.49), so it is analysed relatively well. Against other systems, it is below the lower quartile of the recall and precision scores (medians 0.64 and 0.76 respectively).

- **Key chain 0, `K_0`**

```
<COREF-18> := L    {COREF-1200002}        sc R:1 P:1 => r:1/3 p:1/1
   KEY_CHAIN:           K_0
   OWN_CHAIN:           L_5
## TEXT:               "MR. BARRETT"
                 XX  L= BARRETT

<COREF-19> := L    {COREF-1200001}        sc R:1 P:1 => r:1/3 p:1/1
   KEY_CHAIN:           K_0
   OWN_CHAIN:           L_5
```

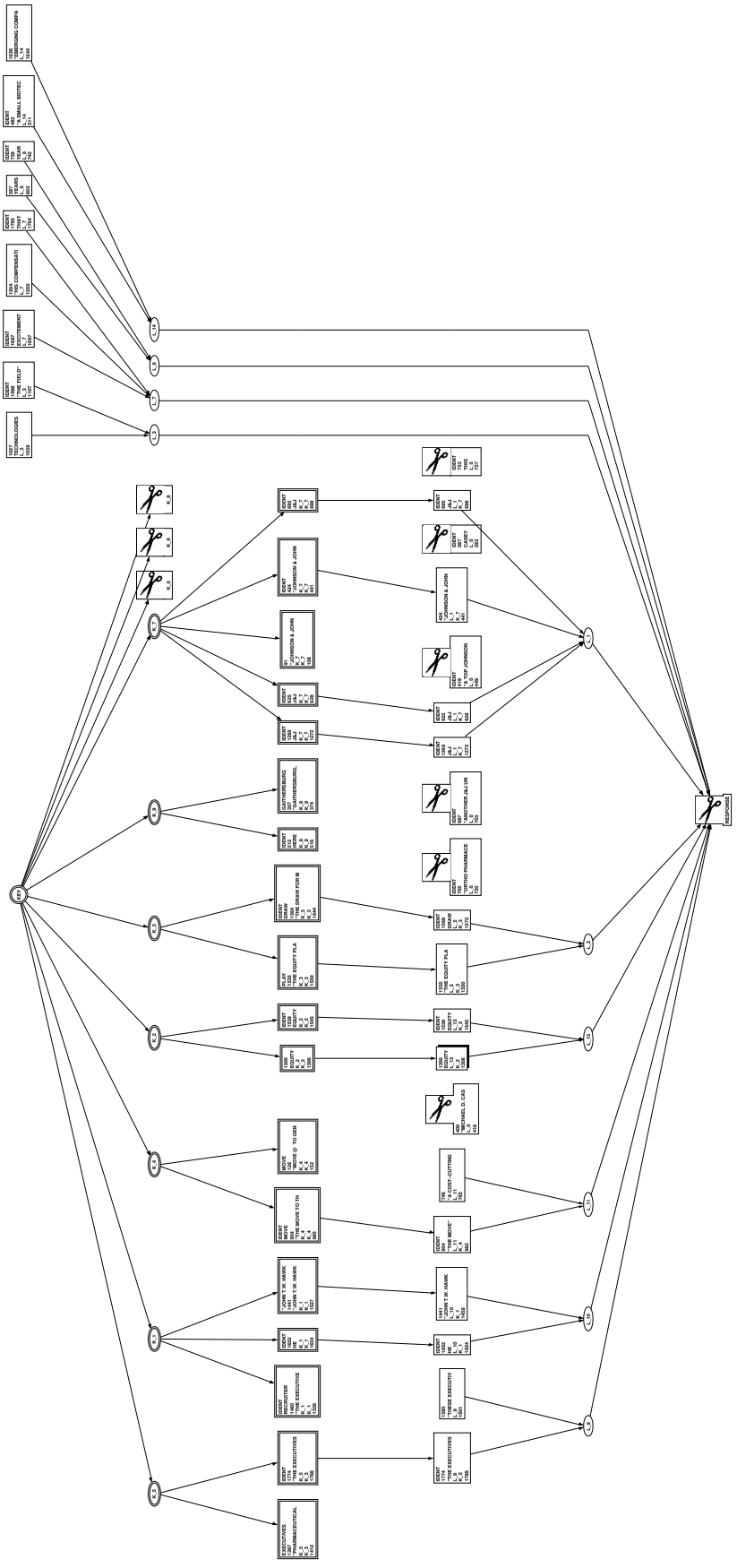


Figure 4.1: Coref key-to-response mapping graph, part one.

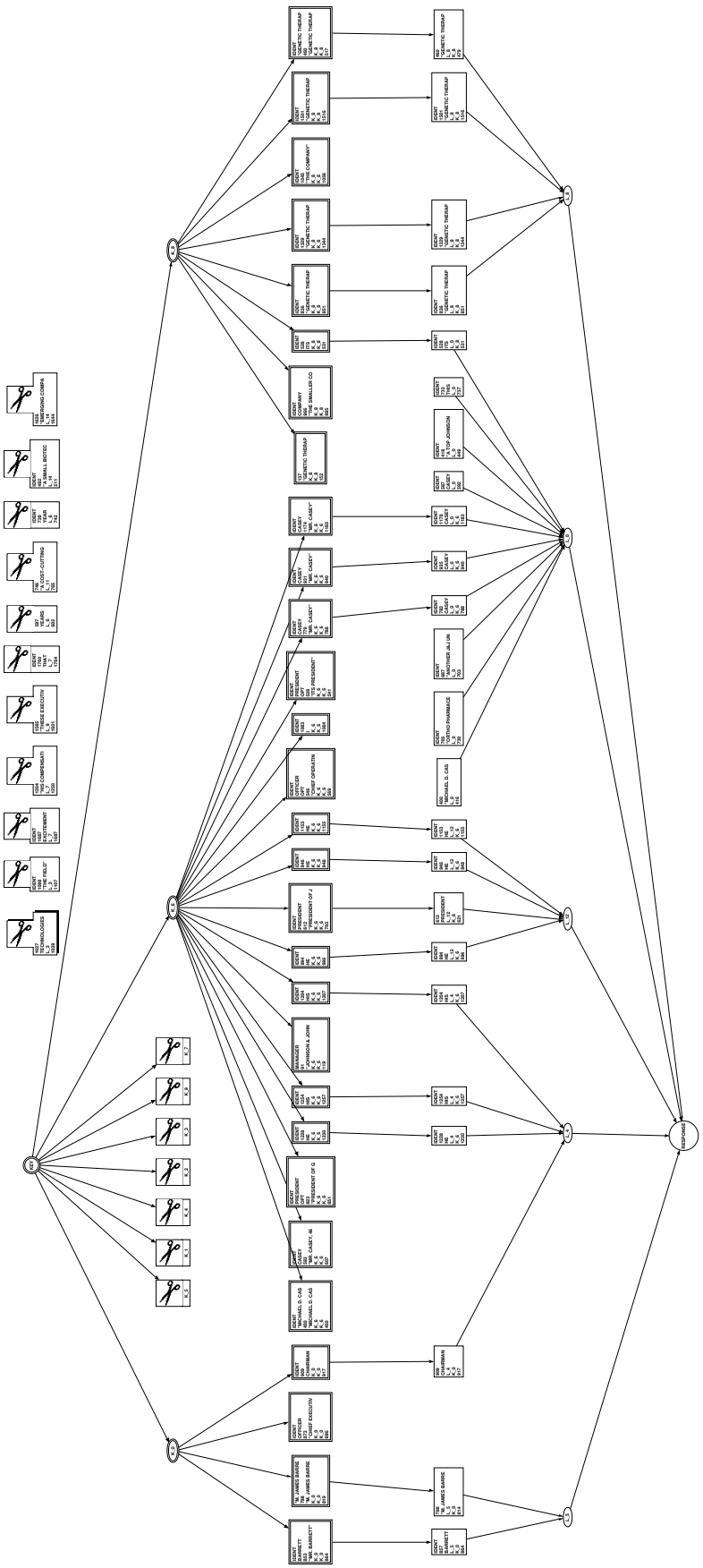


Figure 4.2: Coref key-to-response mapping graph, part two.

```
## TEXT:                "M. JAMES BARRETT, 50,"
                 XX  L= "M. JAMES BARRETT"

<COREF-20> := L    {MISMATCHED OR UNDERGENERATED}   {ignoring...}
   KEY_CHAIN:           K_0
   TEXT:                "CHIEF EXECUTIVE OFFICER"



<COREF-21> := L    {COREF-1000001}        sc XX L_4 scoring in K_6
   KEY_CHAIN:           K_0
   OWN_CHAIN:           L_4
   TEXT:                CHAIRMAN
```

The first two items are coreferenced on the basis of surname. LOLITA correctly analyses the event of Barrett remaining the CEO, but the coref recognising rules do not pick up this information: "to remain" is not understood as indicating sameness. The same happens for Barrett becoming a chairman, but the concept of this chairman is erroneously attached to by the pronouns (in K_6) referring to Casey two paragraphs later. This appears to be due to 'Casey' being known to LOLITA as a *female* name (despite it being more common as a male name), so the pronouns cannot link to it, hence selection of this chairman as the most recent male entity. LOLITA does not distinguish forenames or surnames.

All systems link the first two items, and only one system links all four, though it loses precision by asserting a link between Barrett and "President of Genetic Therapy", which was true at one time, and hence seems allowed by the task specification ([DARPA, 1995] p342, line 13). This would allow Casey and Barrett to be coreferenced as part of the same equivalence class, which is clearly wrong. The author believes this is an example of inconsistency in the task specification. The particular system does not assert that Casey is the president.

- **Key chain 1** (K_1)

```
<COREF-38> := L    {COREF-800001}         sc R:1 P:1 => r:1/2 p:1/1
   KEY_CHAIN:           K_1
   OWN_CHAIN:           L_10
## TEXT:                "JOHN T.W. HAWKINS, THE EXECUTIVE RECRUITER WHO...
                 XX  L= "JOHN T.W. HAWKINS"

<COREF-44> := L    {COREF-800002}         sc R:1 P:1 => r:1/2 p:1/1
   KEY_CHAIN:           K_1
   OWN_CHAIN:           L_10
   TEXT:                HE

<COREF-37> := L    {MISMATCHED OR UNDERGENERATED}   {ignoring...}
   KEY_CHAIN:           K_1
   TEXT:                "THE EXECUTIVE RECRUITER WHO ARRANGED THE...
```

The anaphor is easily resolved, with Hawkins being the most recently mentioned male in the text. The appositional phrase is in itself correctly parsed as referring to Hawkins, and the semantic analysis of the name and phrase appears correct, but they are never linked together. The author cannot determine why this happens.

Two systems get this chain completely correct, and three systems score nothing. Another system gets only the link that LOLITA misses, losing precision in K_6 by then linking to 'Casey'. Interestingly, all systems attempt a markup for 'he'; three fail to score by linking it only to markups of other chains.

- **Key chain 2** (K_2)

```
<COREF-40> := L    {COREF-1500002}        sc R:1 P:1 => r:1/1 p:1/1
   KEY_CHAIN:           K_2
   OWN_CHAIN:           L_13
   TEXT:                EQUITY

<COREF-41> := L    {COREF-1500001}        sc R:1 P:1 => r:1/1 p:1/1
   KEY_CHAIN:           K_2
   OWN_CHAIN:           L_13
   TEXT:                EQUITY
```

The two occurrences of the word 'equity' are linked through one sense of equity. In particular, of the three known concepts of equity, the one for fairness is chosen (sense 3 in WordNet). Four systems get this chain correct.

- **Key chain 3** (K_3)

```
<COREF-42> := L    {COREF-700002}         sc R:1 P:1 => r:1/1 p:1/1
   KEY_CHAIN:           K_3
   OWN_CHAIN:           L_2
## TEXT:               "THE DRAW FOR MANY OF THESE EXECUTIVES IN...
                 XX  L= DRAW

<COREF-43> := L    {COREF-700001}         sc R:1 P:1 => r:1/1 p:1/1
   KEY_CHAIN:           K_3
   OWN_CHAIN:           L_2
   TEXT:                "THE EQUITY PLAY"
```

This link is recognised by the use of an "is a" construction. Although LOLITA coreferenced both of the phrases "draw" and "draw for many of these executives" to the concept for "equity play", the smaller string is always chosen as it avoids possible errors from misparsing (ie, making errors on the scope of a phrase). Two other systems get this chain correct.

- **Key chain 4** (K_4)

```
<COREF-24> := L    {COREF-1100002}     sc XX L_11 not scoring anywhere
   KEY_CHAIN:           K_4
   OWN_CHAIN:           L_11
## TEXT:               "THE MOVE TO THE SMALLER COMPANY"
                 XX  L= "THE MOVE"

<COREF-25> :=
        L    {MISMATCHED OR UNDERGENERATED}   {ignoring...}
   KEY_CHAIN:           K_4
   START OFFSET:        126
   TEXT:                "MOVE @  TO GENETIC THERAPY"

<MISMATCHED or OVERGENERATED> := L-{COREF-1100001}  L_11 appearing in  K_4
?? TEXT:                "A COST-CUTTING MOVE"
?? OWN_CHAIN:           L_11
```

This association is not made because of LOLITA's weak HL (headline) handling: effectively, only phrases that appear as proper nouns in the main text are examined in the headline for possible matches. The L_11 chain links "the move to the smaller company" to "a cost-cutting move" which appears in a previous paragraph: the mistake is in misinterpreting the sense of 'move'. (NB: the other part of L_11 has been added to this key chain for convenience of reference). Only one system gets this link, and four – including that system - make the same mistake as LOLITA in coreferencing to "a cost-cutting move".

- **Key chain 5** (K_5)

```
<COREF-45> := L    {COREF-600002}       sc XX L_9 not scoring anywhere
   KEY_CHAIN:           K_5
   OWN_CHAIN:           L_9
   TEXT:                "THE EXECUTIVES"

<COREF-46> := L    {MISMATCHED OR UNDERGENERATED}   {ignoring...}
   KEY_CHAIN:           K_5
   TEXT:                "PHARMACEUTICAL EXECUTIVES"

<MISMATCHED or OVERGENERATED> := L-{COREF-600001}  L_9 appearing in  K_5
?? TEXT:                "THESE EXECUTIVES"
?? OWN_CHAIN:           L_9
```

"Pharmaceutical executives" is analysed as expected. "The executives" is not coreferenced to the previous concept: LOLITA will not make such links. The marking of "these executives" and its association with "the executives" is an error, as it is part of the larger phrase "many of these executives", which indicates a *subset* of the executives under discussion. Two systems are fully correct here, and three systems get the required link but make the same mistake as LOLITA.

- **Key chain 6** (K_6)

```
<COREF-15> := L    {COREF-300004}         sc R:3 P:3 => r:8/16 p:3/9
   KEY_CHAIN:           K_6
   OWN_CHAIN:           L_0
## TEXT:               "MR. CASEY"
                  XX  L= CASEY

<COREF-22> := L    {COREF-300005}         sc R:3 P:3 => r:8/16 p:3/9
   KEY_CHAIN:           K_6
   OWN_CHAIN:           L_0
## TEXT:               "MR. CASEY"
                  XX  L= CASEY

<COREF-31> := L    {COREF-300006}         sc R:3 P:3 => r:8/16 p:3/9
   KEY_CHAIN:           K_6
   OWN_CHAIN:           L_0
## TEXT:               "MR. CASEY"
                  XX  L= CASEY

<COREF-0> := L     {COREF-300001}         sc R:3 P:3 => r:8/16 P:3/9
   KEY_CHAIN:           K_6
   OWN_CHAIN:           L_0
## TEXT:               "MICHAEL D. CASEY, A TOP JOHNSON & JOHNSON MANAGER
                  XX  L= "MICHAEL D. CASEY"

<COREF-11> := L    {MISMATCHED OR UNDERGENERATED}   {ignoring...}
   END OFFSET:          607
   START OFFSET:        583
   KEY_CHAIN:           K_6
   MIN:                 CASEY
   TEXT:                "MR. CASEY, 46 YEARS OLD,"

<MISMATCHED or OVERGENERATED> := L-{COREF-300003}  L_0 scoring in K_6
?? START OFFSET:        587
?? TEXT:                CASEY
?? OWN_CHAIN:           L_0
?? END OFFSET:          592
```

We will discuss this large key chain by the fragments of response chains which appear in it: L_0, L_4, L_12. L_0 has a bad precision, so we discuss the scoring markups here and discuss the unmapped remainder later. The four scoring markups all mention Casey. These

references are straightforward, as discussed in the NE section, since the full name comes first and subsequent occurrences of the surname match easily to the prior concept. The last markup pair appears to be a scorer error: they clearly overlap (note the offsets) and they should be mapped. This would increase recall and precision for the K_6 and L_0 intersection. Another system is penalised this way. Five systems get all of this portion of K_6. That is, they each produce the four links required. The majority mark the minimum string.

```
<COREF-33> := L    {COREF-1000003}        sc R:2 P:2 => r:8/16 p:2/3
   KEY_CHAIN:           K_6
   OWN_CHAIN:           L_4
   TEXT:                HE

<COREF-34> := L    {COREF-1000004}        sc R:2 P:2 => r:8/16 p:2/3
   KEY_CHAIN:           K_6
   OWN_CHAIN:           L_4
   TEXT:                HIS

<COREF-32> := L    {COREF-1000002}        sc R:2 P:2 => r:8/16 p:2/3
   KEY_CHAIN:           K_6
   OWN_CHAIN:           L_4
   TEXT:                HIS
```

The problem of the name Casey being understood as female has been mentioned above, hence this chain is not joined to L_0 above. The pronouns of L_4 occur in a single sentence with no new male entity introduced, so will all be coreferenced to the same male entity from the previous text, and hence implicitly to each other. The markup causing the imprecision is COREF-21, appearing in K_0, 'chairman', which was the last mentioned entity not thought to be definitely female (ie, it has unspecified gender), hence the resolution. All systems link these anaphora together.

```
<COREF-23> := L    {COREF-1300002}        sc R:3 P:3 => r:8/16 p:3/3
   KEY_CHAIN:           K_6
   OWN_CHAIN:           L_12
   TEXT:                HE

<COREF-30> := L    {COREF-1300004}        sc R:3 P:3 => r:8/16 p:3/3
   KEY_CHAIN:           K_6
   OWN_CHAIN:           L_12
   TEXT:                HE

<COREF-27> := L    {COREF-1300003}        sc R:3 P:3 => r:8/16 p:3/3
   KEY_CHAIN:           K_6
   OWN_CHAIN:           L_12
   TEXT:                HE

<COREF-12> := L    {COREF-1300001}        sc R:3 P:3 => r:8/16 p:3/3
   KEY_CHAIN:           K_6
   OWN_CHAIN:           L_12
## TEXT:               "PRESIDENT OF J&J'S MCNEIL PHARMACEUTICAL SUBSID...
               XX  L= PRESIDENT
```

In L_12, a similar situation with pronouns applies: the pronouns are in a single sentence and hence are all resolved to the same entity. This entity is the 'president' that Casey was. It is not clear why president should be chosen for these pronouns, in preference to the chairman of L_4, though it is the correct answer. Note that this response chain has no imprecision. All systems but one correctly link the anaphors, but only four systems get the link to 'President'.

```
<COREF-16> := L    {MISMATCHED OR UNDERGENERATED}   {ignoring...}
   KEY_CHAIN:           K_6
   STATUS:              OPT
   TEXT:                "PRESIDENT OF GENETIC THERAPY"
```

The concept of this president is built, and the event of Casey succeeding Barrett, but the relationship between these facts is not determined by LOLITA. Thus, the inference of Casey being president is never made. Only one system makes this link, although one system links this markup to K_0 as noted before.

```
<COREF-10> := L    {MISMATCHED OR UNDERGENERATED}   {ignoring...}
   KEY_CHAIN:           K_6
   STATUS:              OPT
   MIN:                 OFFICER
   TEXT:                "CHIEF OPERATING OFFICER"

<COREF-9> := L    {MISMATCHED OR UNDERGENERATED}   {ignoring...}
   KEY_CHAIN:           K_6
   STATUS:              OPT
   MIN:                 PRESIDENT
   TEXT:                "ITS PRESIDENT"
```

The phrase "its president and chief operating officer" is mis-parsed (giving an interpretation along the lines of a "(president and chief) who operates an officer"), and then this concept is analysed as the thing Casey will become. The two components are not treated as separate concepts because of the wrong bracketing in parsing, so are not separately markable. Furthermore, 'to become' is not a verb that LOLITA used at the time to identify coreferences. Only two systems make this connection.

```
<COREF-1> := L    {MISMATCHED OR UNDERGENERATED}   {ignoring...}
   KEY_CHAIN:           K_6
   MIN:                 MANAGER
   TEXT:                "JOHNSON & JOHNSON @  MANAGER"
```

LOLITA's title analysis only considers proper name matches, hence does not link the title occurrence of this phrase to its occurrence in the main text. Only one system is correct here, although it is imprecise in adding 'a top Johnson' and '& Johnson Manager' to its chain. Incidentally, this chain also includes the two previous markups but none other.

```
<COREF-29> := L    {MISMATCHED OR UNDERGENERATED}   {ignoring...}
   KEY_CHAIN:           K_6
   TEXT:                I
```

LOLITA's handling of reported speech does not take the speaker's identity in to account, hence 'I' is resolved to its default speaker, R. Garigliano. Only one other system fails to score on this markup.

Overall, all systems do better than LOLITA in this chain. The best answer on this chain misses two links in this chain and over-generates four links to lose on precision.

- **Key chain 7** (K_7)

```
<COREF-35> := L    {COREF-500004}         sc R:3 P:3 => r:3/4 p:3/3
   KEY_CHAIN:           K_7
   OWN_CHAIN:           L_1
   TEXT:                J&J

<COREF-14> := L    {COREF-500003}         sc R:3 P:3 => r:3/4 p:3/3
   KEY_CHAIN:           K_7
   OWN_CHAIN:           L_1
   TEXT:                J&J

<COREF-2> := L    {COREF-500001}         sc R:3 P:3 => r:3/4 p:3/3
   KEY_CHAIN:           K_7
   OWN_CHAIN:           L_1
```

```
   TEXT:              "JOHNSON & JOHNSON"

<COREF-13> := L    {COREF-500002}        sc R:3 P:3 => r:3/4 p:3/3
   KEY_CHAIN:           K_7
   OWN_CHAIN:           L_1
   TEXT:                J&J

<COREF-3> := L    {MISMATCHED OR UNDERGENERATED}   {ignoring...}
   KEY_CHAIN:           K_7
   TEXT:                "JOHNSON & JOHNSON"
```

As explained for NE, the occurrences of 'J&J' are attached to the previous occurrence of the full name. The missed markup is from the headline: although this is marked as a named entity, LOLITA fails to link it to any of the text occurrences. Two systems have perfect recall, although one loses perfect precision by coreferencing to 'the smaller company' of K_8. Two systems fail to score.

- **Key chain 8** (K_8)

```
<COREF-4> := L    {COREF-400001}        sc R:3 P:3 => r:3/7 p:3/3
   KEY_CHAIN:           K_8
   OWN_CHAIN:           L_8
## TEXT:               "GENETIC THERAPY INC., A SMALL BIOTECHNOLOGY...
                 XX  L= "GENETIC THERAPY INC"

<COREF-36> := L    {COREF-400003}        sc R:3 P:3 => r:3/7 p:3/3
   KEY_CHAIN:           K_8
   OWN_CHAIN:           L_8
   TEXT:                "GENETIC THERAPY"

<COREF-17> := L    {COREF-400002}        sc R:3 P:3 => r:3/7 p:3/3
   KEY_CHAIN:           K_8
   OWN_CHAIN:           L_8
   TEXT:                "GENETIC THERAPY"

<COREF-39> := L    {COREF-400004}        sc R:3 P:3 => r:3/7 p:3/3
   KEY_CHAIN:           K_8
   OWN_CHAIN:           L_8
   TEXT:                "GENETIC THERAPY"

<COREF-5> := L    {MISMATCHED OR UNDERGENERATED}   {ignoring...}
   KEY_CHAIN:           K_8
   TEXT:                "GENETIC THERAPY"
```

These are straightforward as they are identical proper names, allowing for the 'Inc' which can occur in company names. LOLITA's missed markup occurs in the headline. Four systems get all four links in this portion.

```
<COREF-8> := L    {COREF-300002}        sc XX L_0 scoring in K_6
   KEY_CHAIN:           K_8
   OWN_CHAIN:           L_0
   TEXT:                ITS
```

This possessive pronoun is erroneously resolved to Casey. It is a consequence of LOLITA's method of handling infinitive constructions, which introduces a new reference as the subject of the verb. This is the most recent compatible referent when the determiner is encountered. One other system makes the wrong linkage, two fail to produce this markup, and the other three link it to other markups in this chain.

```
<COREF-28> := L    {MISMATCHED OR UNDERGENERATED}   {ignoring...}
   KEY_CHAIN:           K_8
```

```
   TEXT:                "THE COMPANY"

<COREF-26> :=
        L    {MISMATCHED OR UNDERGENERATED}   {ignoring...}
   KEY_CHAIN:           K_8
   MIN:                 COMPANY
   TEXT:                "THE SMALLER COMPANY"
```

LOLITA builds these as distinct companies and fails to link them. Four systems make this connection, though the other two link one of these to the correct chain.

- **Key chain 9** (K_9)

```
<COREF-7> := L    {MISMATCHED OR UNDERGENERATED}   {ignoring...}
   KEY_CHAIN:           K_9
   MIN:                 GAITHERSBURG
   TEXT:                "GAITHERSBURG, MD."

<COREF-6> := L    {MISMATCHED OR UNDERGENERATED}   {ignoring...}
   KEY_CHAIN:           K_9
   TEXT:                HERE
```

'Here' is interpreted literally, defaulting to the office where R. Garigliano works. Hence the coreference is never made to the location in the dateline. No system produces markups for this chain.

- **Unmapped responses – remainder of L_0**

```
<MISMATCHED or OVERGENERATED> := L-{COREF-300007}  L_0 scoring in K_6
?? TEXT:                "A TOP JOHNSON & JOHNSON MANAGER"
?? OWN_CHAIN:           L_0
```

This appositional phrase is *indefinite*, which is not markable in the final task specification. Thus, LOLITA lacks this filter. One system makes a similar mistake, another a similar (maybe worse) mistake of producing two markups 'a top Johnson' and '& Johnson manager'.

```
<MISMATCHED or OVERGENERATED> := L-{COREF-3000010}  L_0 scoring in K_6
?? TEXT:                "ORTHO PHARMACEUTICAL CORP"
?? OWN_CHAIN:           L_0

<MISMATCHED or OVERGENERATED> := L-{COREF-300008}  L_0 scoring in K_6
?? TEXT:                "ANOTHER J&J UNIT"
?? OWN_CHAIN:           L_0

<MISMATCHED or OVERGENERATED> := L-{COREF-300009}  L_0 scoring in K_6
?? TEXT:                THIS
?? OWN_CHAIN:           L_0
```

The first two markups are incorrect, because the appositional phrase (the second markup) is indefinite and should not be marked, hence it is not a valid coreferent for the company, which has no other coreferents. The phrase 'this year' is mis-parsed, resulting in a splitting of the phrase and the determiner being identified as a demonstrative pronoun, which would then need to be resolved with some concept, the most suitable being 'another J&J unit', hence that link.

These incorrect markups are coreferenced to Casey. This is due to a semantics error which interprets "Casey is president of X" as "Casey is X". A parsing error results in X being "Ortho Pharmaceutical Corp" and "another J&J unit". The final coreference recognition produces links between the subjects and objects of 'is_a' events.

One system produces the first markup: in fact, it marks that phrase twice and joins it to different chains. Four systems produce the 'another unit' phrase, two linked to scoring chains, two not, and one produces the full phrase which contains the two. No system produces 'this' as a markup.

- **Unmapped response chain L_3**

```
<MISMATCHED or OVERGENERATED> := L-{COREF-900001}  L_3 not linked anywhere
?? TEXT:                TECHNOLOGIES
?? OWN_CHAIN:           L_3

<MISMATCHED or OVERGENERATED> := L-{COREF-900002}  L_3 not linked anywhere
?? TEXT:                "THE FIELD"
?? OWN_CHAIN:           L_3
```

This link occurs because of the similar meaning, but is incorrect because of the differing quantification, ie "technologies like X", but X is "the field". LOLITA is alone in producing this chain.

- **Unmapped response chain L_14**

```
<MISMATCHED or OVERGENERATED> := L-{COREF-100002}  L_14 not linked anywhere
?? TEXT:                "A SMALL BIOTECHNOLOGY CONCERN"
?? OWN_CHAIN:           L_14

<MISMATCHED or OVERGENERATED> := L-{COREF-100001}  L_14 not linked anywhere
?? TEXT:                "EMERGING COMPANIES"
?? OWN_CHAIN:           L_14
```

This appears to be a bug in the reference resolution. The two phrases appear wide apart in the article, in the order shown above. Three systems link the first markup to 'Genetic Therapy' in K_8, which is illegal as it is an indefinite appositional phrase. No other system produces the second markup.

- **Unmapped response chain L_6**

```
<MISMATCHED or OVERGENERATED> := L-{COREF-1400002}  L_6 not linked anywhere
?? TEXT:                YEAR
?? OWN_CHAIN:           L_6

<MISMATCHED or OVERGENERATED> := L-{COREF-1400001}  L_6 not linked anywhere
?? TEXT:                YEARS
?? OWN_CHAIN:           L_6
```

The occurrences of these words are linked to the basic concepts, but they are not filtered out whilst calculating coref markups. This mistake is specific to LOLITA.

- **Unmapped response chain L_7**

```
<MISMATCHED or OVERGENERATED> := L-{COREF-200003}  L_7 not linked anywhere
?? TEXT:                EXCITEMENT
?? OWN_CHAIN:           L_7

<MISMATCHED or OVERGENERATED> := L-{COREF-200002}  L_7 not linked anywhere
?? TEXT:                THAT
?? OWN_CHAIN:           L_7

<MISMATCHED or OVERGENERATED> := L-{COREF-200001}  L_7 not linked anywhere
?? TEXT:                "HIS COMPENSATION, WHICH HE SAID WAS COMPARABLE"
?? OWN_CHAIN:           L_7
```

A parsing error for 'that' makes it a demonstrative pronoun, subsequently resolved to 'his compensation'. Why this resolution should happen is not clear – it is definitely a bug. Another parsing error, similar to the one that lead to coreferencing Casey and his company, associates 'excitement' and the resolved entity for 'that'. Again, another LOLITA-only error.

- **Observations on the remaining markups of other systems**

  These are notable points from the unmapped templates, which were seen in several systems. Six systems mark the 'it' of "it's the excitement...", two linking it to `K_3` and three to `K_8`. Four mark the noun of "emerging companies" (LOLITA marks the complete phrase), but do not link it to anything useful. Four mark "(small) biotech companies". The high degree of replication is interesting, especially with the 'it' - which may indicate surface anaphora resolution techniques.

Despite the help of the comparison tool, coreference results are not easy to analyse. The relationships between markup chains are sometimes too complex to represent in a linear form as above. The significance of precision scores is also hard to grasp, as the display is necessarily key-oriented. However, this alternative way of viewing coref results is a vast improvement over raw SGML text, so the author concludes this effort is reasonably successful. The coref mode of the comparison tool could be improved by suggestions from regular users.

Another method of looking at coreference performance is by considering the type of the markup. This approach is being investigated in Durham by Urbanowicz[3]. A type such as "abbreviated name" or "NP with indefinite article" is added to each SGML markup. In scoring, all possible pairs in all chains in the key and response are calculated. Recall points are given for the types in each response pair if both markups in a pair are mapped to a pair in the key. Precision points are deducted if this is not the case. The result is a breakdown of the coref scores in terms of the types involved in links. For this article, we get the following: briefly, the major contributors to the score – names, possessive determiners, and pronouns – are equally weak. All of the smaller number of acronyms are found, but with a little imprecision.

	RESP	KEY	OVG	REC	PRE
(ABNM) Abbreviated name:	48	95	6	0.51	0.89
(FLNM) Full name:	23	37	5	0.62	0.82
(ACNM) Acronym:	12	12	3	1.00	0.80
(APNN) Inside appositive:	0	2	0	undef	-
(PRNN) Premod in compounds:	2	2	2	1.00	0.50
(TRNN) NP inside ternary copula:	2	16	0	0.12	1.00
(DANN) NP with definite article:	0	18	0	undef	-
(PSNN) NP with possessive det:	2	16	0	0.12	1.00
(NDNN) Sing noun with no det:	2	55	0	0.04	1.00
(NDPN) Plural noun with no det:	0	1	0	undef	-
(PSDT) Possessive det:	23	39	4	0.59	0.85
(PRPR) Ordinary pronoun:	36	83	13	0.43	0.73
Totals for markable objects:	150	376	33		

Spuriously marked: 29

4.4 Template Element

This summary of the differences between the key and LOLITA's response was generated by the template comparison tool (see appendix B for an explanation of the symbols used). Recall (at 55%) is just above median, and precision (75%) is just above the upper quartile of LOLITA results. Against the other eleven systems, LOLITA is just above the lower quartile of precision and is a low outlier for the recall (ie, it is significantly below the main group). The output below is from the multi-system comparison. It is included for TE as it is relatively short compared to the equivalent for other tasks. LOLITA is system '`a`'.

- **Organization-1: Johnson & Johnson**

[3] email A.J.Urbanowicz@durham.ac.uk for further details.

```
<ORGANIZATION-1> :=
        b    {ORGANIZATION-2}       sc 40.00
        c    {ORGANIZATION-5}       sc 66.67
        d    {ORGANIZATION-1}       sc 66.67
        e    {ORGANIZATION-1}       sc 66.67
        f    {ORGANIZATION-18}      sc 40.00
        g    {ORGANIZATION-3}       sc 100.00
        h    {ORGANIZATION-1}       sc 100.00
        i    {ORGANIZATION-3}       sc 100.00
        j    {ORGANIZATION-1}       sc 66.67
        k    {ORGANIZATION-2}       sc 85.71
        a    {MISMATCHED OR UNDERGENERATED}   {ignoring...}
        l    {MISMATCHED OR UNDERGENERATED}   {ignoring...}
## ORG_ALIAS:           J&J
                ==      d e g h i k
                XX  j= Johnson
                --      b c f
++ ORG_DESCRIPTOR:
                ++  k= the smaller company
                ++  c= an emerging pharmaceutical company
                ++  f= an emerging pharmaceutical company
## ORG_NAME:            Johnson & Johnson
                ==      c g h i j k
                XX  b= J&J
                XX  d= J&J's McNeil Pharmaceutical
                XX  e= J&J's McNeil Pharmaceutical
                --      f
   ORG_TYPE:            COMPANY

<MISMATCHED or OVERGENERATED> := L-{PERSON-96562}
?? PER_ALIAS:           Genetic Therapy : J&J
?? PER_NAME:            Johnson & Johnson
```

These two templates are not matched. As noted in NE, "Johnson & Johnson" is mis-typed as a person, so consequently a PERSON template is produced by LOLITA. An additional error is the (internal) coreferencing with "Genetic Therapy" from the title – which is also coreferenced to the main text occurrences of the name. Its cause appears to be a misinterpretation of the first sentence. This mistake does not show in the coref task: there appears to be a bug in the textref marking scheme which prevents that erroneous coref markup from being generated. The template tasks use different mechanisms for creating slot fills.

System l makes a similar mistake, by creating a person template for 'Johnson'. Three systems over-generate an ORG_DESCRIPTOR. Three are incorrect in the ORG_NAME (b,d,e), and a fourth (f) omits it. One of the three produces the alias instead of the name required by the task specification. All identify the type correctly.

- **Organization-2: Genetic Therapy Inc**

```
<ORGANIZATION-2> :=
        a    {ORGANIZATION-96580}   sc 66.67
        b    {ORGANIZATION-5}       sc 60.00
        c    {ORGANIZATION-4}       sc 60.00
        d    {ORGANIZATION-4}       sc 60.00
        e    {ORGANIZATION-4}       sc 66.67
        f    {ORGANIZATION-20}      sc 60.00
        g    {ORGANIZATION-1}       sc 60.00
        h    {ORGANIZATION-2}       sc 66.67
        i    {ORGANIZATION-1}       sc 60.00
        j    {ORGANIZATION-2}       sc 72.73
        k    {ORGANIZATION-3}       sc 60.00
        l    {ORGANIZATION-1}       sc 66.67
   ORG_NAME:            Genetic Therapy Inc.
   ORG_TYPE:            COMPANY
## ORG_ALIAS:           Genetic Therapy
                ==      a b c d e f g h i j l
                >>  k= Genetic Therapy : Therapy
## ORG_DESCRIPTOR:      a small biotechnology concern here
```

```
                    / the smaller company
                XX  g= the company
                XX  d= pharmaceutical company
                XX  k= J&J's McNeil Pharmaceutical subsidiary
                >>  j= a small biotechnology concern : the smaller company
                XX     a small biotechnology concern
                    XX i b c f
                --     a e h l
## ORG_COUNTRY:        United States
                --     a b c d e f g h i j k l
## ORG_LOCALE:         GAITHERSBURG CITY
                --     a b c d e f g h i j k l
```

In common with the other systems, LOLITA gets name and type correct but omits the country and locale. It gets the alias right, and omits the descriptor. The reasons for getting the name and alias correct have been discussed in the sections above on NE and CO. The type defaults to COMPANY in absence of other information. Currently, a country will only be produced if there is a locale, and since the dateline is not used properly, there is no locale.

Descriptors are produced by examining the textrefs of the organisation node and concepts identical to it (ie, connected via an 'is a' link). Often, many phrases are found but not all are possible descriptors. The set is filtered to remove names and insubstantial phrases like "the company". To limit over-generation, possible descriptors not starting with a determiner are removed: this heuristic has been found correct (by experimentation) for around 90% of the key descriptors in the training articles. Finally, only the largest remaining descriptor is used. Again, this is to limit over-generation. In this article, however, none of the acceptable descriptor fills is attached to a point from where it may be recovered.

It is interesting that no system attempts the country or locale slots. For the descriptor, four systems omit it, three are quite wrong, but five come close though omit the final 'here'. Those five would be correct under the ST key – there is an *inconsistency* between the keys (see section 6.2.3). Note that indefinite noun phrases are valid descriptors in the template tasks, in contrast to the unmarkability of indefinite appositional phrases in Coref. System j correctly produces the second alternative for this slot, hence the slightly higher score.

- **Organization-3: McNeil Pharmaceutical**

```
<ORGANIZATION-3> :=
        a    {ORGANIZATION-96650}   sc 80.00
        b    {ORGANIZATION-3}       sc 85.71
        c    {ORGANIZATION-3}       sc 80.00
        f    {ORGANIZATION-4}       sc 80.00
        g    {ORGANIZATION-2}       sc 40.00
        h    {ORGANIZATION-3}       sc 80.00
        i    {ORGANIZATION-4}       sc 80.00
        j    {ORGANIZATION-3}       sc 40.00
        d    {MISMATCHED OR UNDERGENERATED}   {ignoring...}
        e    {MISMATCHED OR UNDERGENERATED}   {ignoring...}
        k    {MISMATCHED OR UNDERGENERATED}   {ignoring...}
        l    {MISMATCHED OR UNDERGENERATED}   {ignoring...}
## ORG_DESCRIPTOR:      subsidiary
                ==      b
                XX  g= pharmaceutical company
                --      a c f h i j
++ ORG_LOCALE:          J&J UNKNOWN
                ++      b
## ORG_NAME:            McNeil Pharmaceutical
                ==      a b c f h i j
                --      g
## ORG_TYPE:            COMPANY
                ==      a b c f g h i
                XX  j= OTHER
```

LOLITA misses only the descriptor, on account of the "must start with determiner" heuristic. The name is easily recognised by the capitalisation, and the type is assigned by default.

The internal analysis of "J&J's McNeil Pharmaceutical subsidiary" is close to correct, but makes the error of linking to a generic concept of subsidiary, rather than the particular one. However, only one system does get this correct, and the remaining slots too, though with one over-generation. Five systems produce identical results to LOLITA, and four systems miss this template entirely.

- **Organization-4: Ortho Pharmaceutical Corp**

```
<ORGANIZATION-4> :=
        a    {ORGANIZATION-96680}   sc 80.00
        b    {ORGANIZATION-1}       sc 80.00
        c    {ORGANIZATION-1}       sc 66.67
        d    {ORGANIZATION-2}       sc 100.00
        e    {ORGANIZATION-2}       sc 100.00
        f    {ORGANIZATION-5}       sc 66.67
        g    {ORGANIZATION-4}       sc 80.00
        h    {ORGANIZATION-4}       sc 66.67
        i    {ORGANIZATION-2}       sc 80.00
        j    {ORGANIZATION-5}       sc 80.00
        k    {ORGANIZATION-1}       sc 100.00
        l    {ORGANIZATION-3}       sc 80.00
## ORG_DESCRIPTOR:      another J&J unit
                ==      d e k
                XX  c= the smaller company
                XX  f= another J & J unit
                XX  h= this year in a cost-cutting move
                --      a b g i j l
   ORG_NAME:            Ortho Pharmaceutical Corp.
   ORG_TYPE:            COMPANY
```

Apart from the descriptor, this template is easy to fill – the company name from the capitalisation, and the type on the basis of the 'Corp' suffix. All systems produce this template, with three gaining a perfect score. The only difference between systems is on the descriptor. System f loses points by not reproducing the spacing of the source text.

- **Person-1: Michael D. Casey**

```
<PERSON-1> :=
        a    {PERSON-96549}         sc 100.00
        b    {PERSON-4}             sc 100.00
        c    {PERSON-2}             sc 100.00
        d    {PERSON-3}             sc 100.00
        e    {PERSON-3}             sc 100.00
        f    {PERSON-20}            sc 100.00
        g    {PERSON-4}             sc 100.00
        h    {PERSON-1}             sc 100.00
        i    {PERSON-1}             sc 100.00
        j    {PERSON-2}             sc 100.00
        k    {PERSON-3}             sc 100.00
        l    {PERSON-2}             sc 100.00
   PER_ALIAS:           Casey
   PER_NAME:            Michael D. Casey
   PER_TITLE:           Mr.
```

All systems are correct here. LOLITA's coreferencing of the full name and surname has been discussed above. The title is determined by searching through a list of standard titles for one that occurs in the text attached to the concept for the person of this template – thus 'Mr X' must have previously been linked to the concept.

- **Person-2: M. James Barrett**

```
<PERSON-2> :=
        a    {PERSON-96682}          sc 66.67
        b    {PERSON-3}              sc 100.00
        c    {PERSON-1}              sc 80.00
        d    {PERSON-2}              sc 100.00
        e    {PERSON-2}              sc 100.00
        f    {PERSON-13}             sc 100.00
        g    {PERSON-2}              sc 100.00
        h    {PERSON-2}              sc 80.00
        i    {PERSON-2}              sc 100.00
        j    {PERSON-3}              sc 100.00
        k    {PERSON-1}              sc 100.00
        l    {PERSON-3}              sc 100.00
## PER_ALIAS:           Barrett
                ==      a b d e f g h i j k l
                --      c
## PER_NAME:            M. James Barrett
                ==      b c d e f g h i j k l
                XX  a=  James Barrett
## PER_TITLE:           Mr.
                ==      a b c d e f g i j k l
                --      h
```

Only three systems fail to get this completely correct, LOLITA getting the worst mark by supplying the wrong name (the other two omit the alias and title respectively). The name error is caused by not understanding the abbreviated first-name convention coupled with the middle name being a viable fore-name.

- **Person-3: John T.W. Hawkins**

```
<PERSON-3> :=
        b    {PERSON-1}              sc 100.00
        d    {PERSON-1}              sc 100.00
        e    {PERSON-1}              sc 100.00
        i    {PERSON-3}              sc 100.00
        k    {PERSON-2}              sc 100.00
        l    {PERSON-1}              sc 100.00
        a    {MISMATCHED OR UNDERGENERATED}   {ignoring...}
        c    {MISMATCHED OR UNDERGENERATED}   {ignoring...}
        f    {MISMATCHED OR UNDERGENERATED}   {ignoring...}
        g    {MISMATCHED OR UNDERGENERATED}   {ignoring...}
        h    {MISMATCHED OR UNDERGENERATED}   {ignoring...}
        j    {MISMATCHED OR UNDERGENERATED}   {ignoring...}
   PER_NAME:            John T.W. Hawkins
```

There is only one feature in this template. Six systems, including LOLITA, do not produce this template, though f comes very close but inserts a space between the initials, and g and j omit the surname. LOLITA fails to produce this for the same reason as the NE markup was missed.

- **Points from remaining (unmapped) templates**

  LOLITA's single unmapped template was shown with Organization-1. Two systems produce a spurious organisation "Johnson & Johnson" with type `OTHER`. This is a possible scorer error. Three systems produce a 'J&J' organisation, two of company type and one of other. Two systems produce a person from 'Makes' and 'Makes Move' in the title.

4.5 Scenario Template

As before, the output was produced by the template comparison tool. The templates are discussed in breadth-first order. There does not appear to be an ideal order for discussion, as templates may need reference to both parents and children for interpretation. For example, IN_AND_OUTs relate to the job (and company) expected in its parent, and understanding SUCCESSION_EVENTs

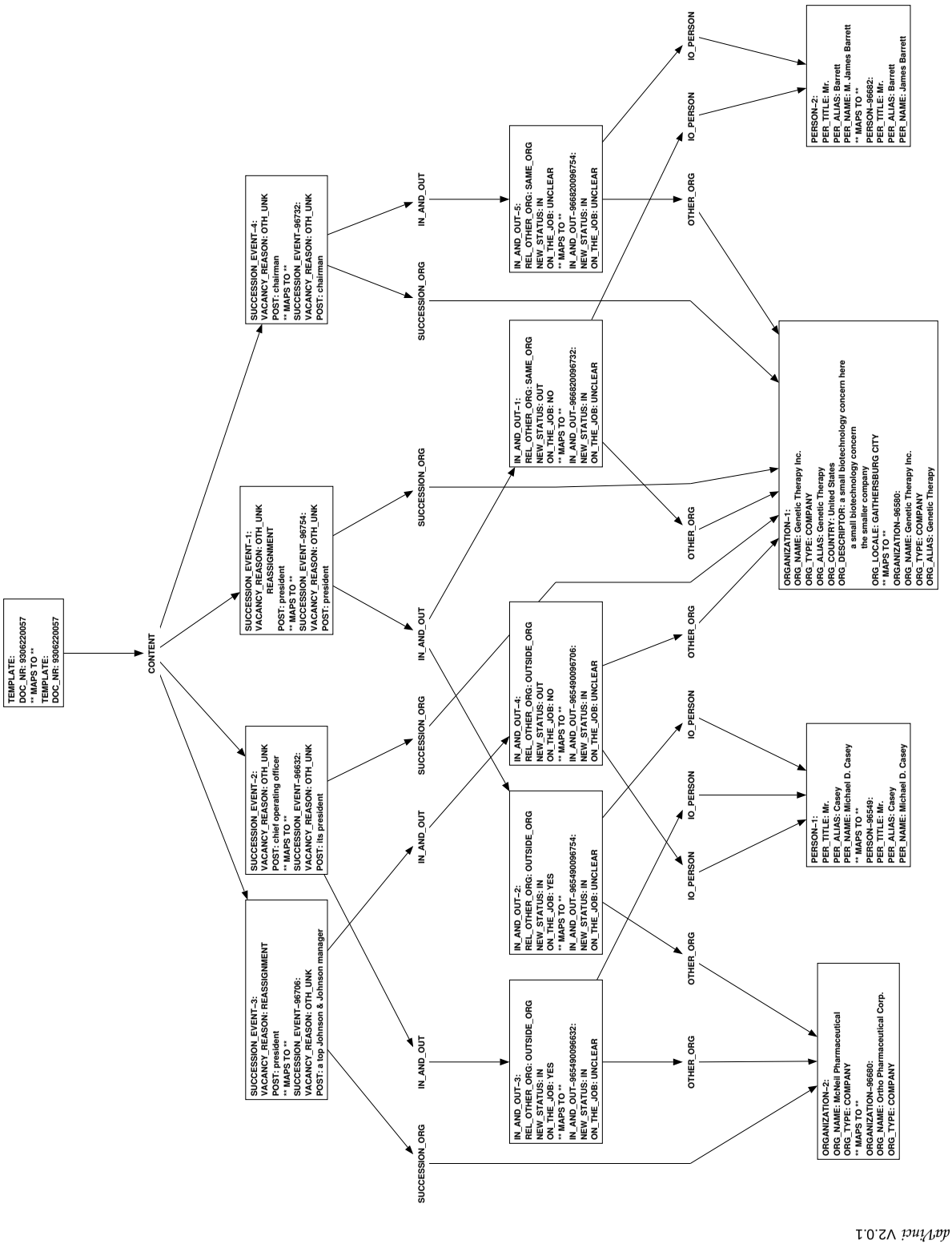


Figure 4.3: Graph of the scenario template key for article 9306220057, indicating mapping to LOLITA templates.

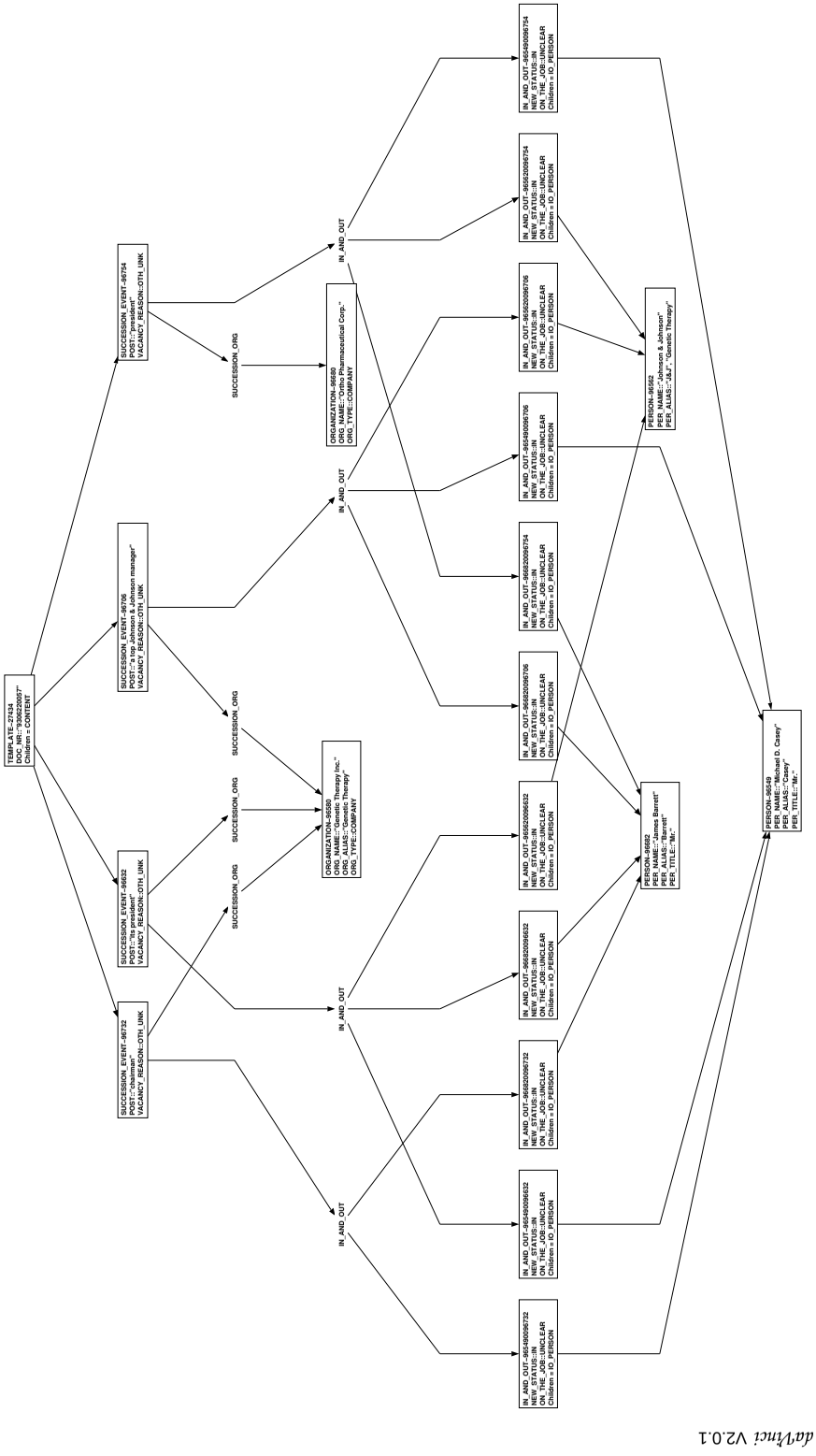


Figure 4.4: (Reduced) Graph of LOLITA's response for article 9306220057.

needs reference to their IN_AND_OUTs. Some of the problems derive from the style of definition of the management succession 'ontology': this matter is pursued in section 6.2.2.

The following precis of the relevant facts from the article is useful in interpretation. Casey (PERSON-1) leaves the job of president in McNeil Pharmaceutical (ORGANIZATION-2). Casey becomes president and chief operating officer of Genetic Therapy (ORGANIZATION-1). Barrett (PERSON-2) leaves the job of president of Genetic Therapy and becomes its chairman (though remains the chief executive officer).

Figure 4.3 shows the key graphically, indicating for each key template the value slots of the response template matched to it as an aid to reference. The graph was extracted automatically from the scenario templates using part of the template comparison tool, and drawn by daVinci [Fröhlich and Werner, 1997]. Figure 4.4 shows an *edited* version of LOLITA's response. There was a high number of irrelevant templates produced due to a bug which builds IN_AND_OUT templates based on R. Garigliano and LOLITA, which obscure the basic answer and decrease the precision score. These are not shown in the graph, but are discussed below.

There are several immediate differences between the graphs. Most noticeably, in the number of IN_AND_OUT templates: LOLITA has eleven compared to five in the key, not including the seven irrelevant ones already removed. Also significant is the involvement of an incorrect PERSON in the IN_AND_OUTs of three SUCCESSION_EVENTs (but not the "chairman" template). Finally, the IN_AND_OUT slots of OTHER_ORG and REL_OTHER_ORG do not appear anywhere: these were not implemented in LOLITA, essentially due to lack of time.

These errors are mainly the result of the simple method of generating templates, as discussed in section 3.2.11: when a succession event is found, LOLITA attempts to create sub-templates based on nodes from the set of nodes within five arcs of the succession event. Only certain kinds of arcs are followed – basically the ones important to events: subjects and objects (and their inverses), plus some minor ones such as destinations. This can improve recall by picking up information which is loosely connected to a succession event (eg, because of weak analysis), but loses on precision because nodes which are a part of different events can allow templates to be wrongly connected to the first event. There is no further check on relevance of a sub-template to its parent. The resulting over-generation is clear in the figure 4.4, when compared to figure 4.3.

Figure 4.5 shows the node pool for event node 96754, which is the basis of the template mapped to SUCCESSION_EVENT-1. It gives an idea of how many nodes are within five arcs of the allowed kind of the main event. The diagram was produced using an interactive semantic net display program written by the author which uses daVinci to visualise the graphs, and was fine-tuned (rearranged) by hand to improve readability. The central succession event is a double-bordered ellipse. Key nodes for the sub-templates are rhombuses. Thick arrows are `object_` arcs, hollow arrows are `subject_` arcs. As an idea of scale, these 72 nodes are approximately a seventh of the 481 new nodes produced in the analysis of this article, not counting the changes to existing nodes, such as adding subsets and instances to existing concepts.

Because answers are generated from a large pool of nodes, and the rules for acceptance are not sufficiently strict, checking the reasons for answers can be a long process and the reasons are rarely interesting. Hence, the explanations below are brief. For the multi-system comparison, it is probable that other systems may also suffer from the inherent problems of mechanically scoring this task (discussed below and in section 6.2.4), so these comments will be brief. More thorough comments would require detailed manual analysis of other systems' output.

- **The top template**

```
<TEMPLATE> :=
        a    {TEMPLATE}              sc 0.00
   DOC_NR:              9306220057
   CONTENT:
        <SUCCESSION_EVENT-1>       a {SUCCESSION_EVENT-96754}     sc 50.00
        <SUCCESSION_EVENT-2>       a {SUCCESSION_EVENT-96632}     sc 50.00
        <SUCCESSION_EVENT-3>       a {SUCCESSION_EVENT-96706}     sc 16.67
        <SUCCESSION_EVENT-4>       a {SUCCESSION_EVENT-96732}     sc 54.55
```

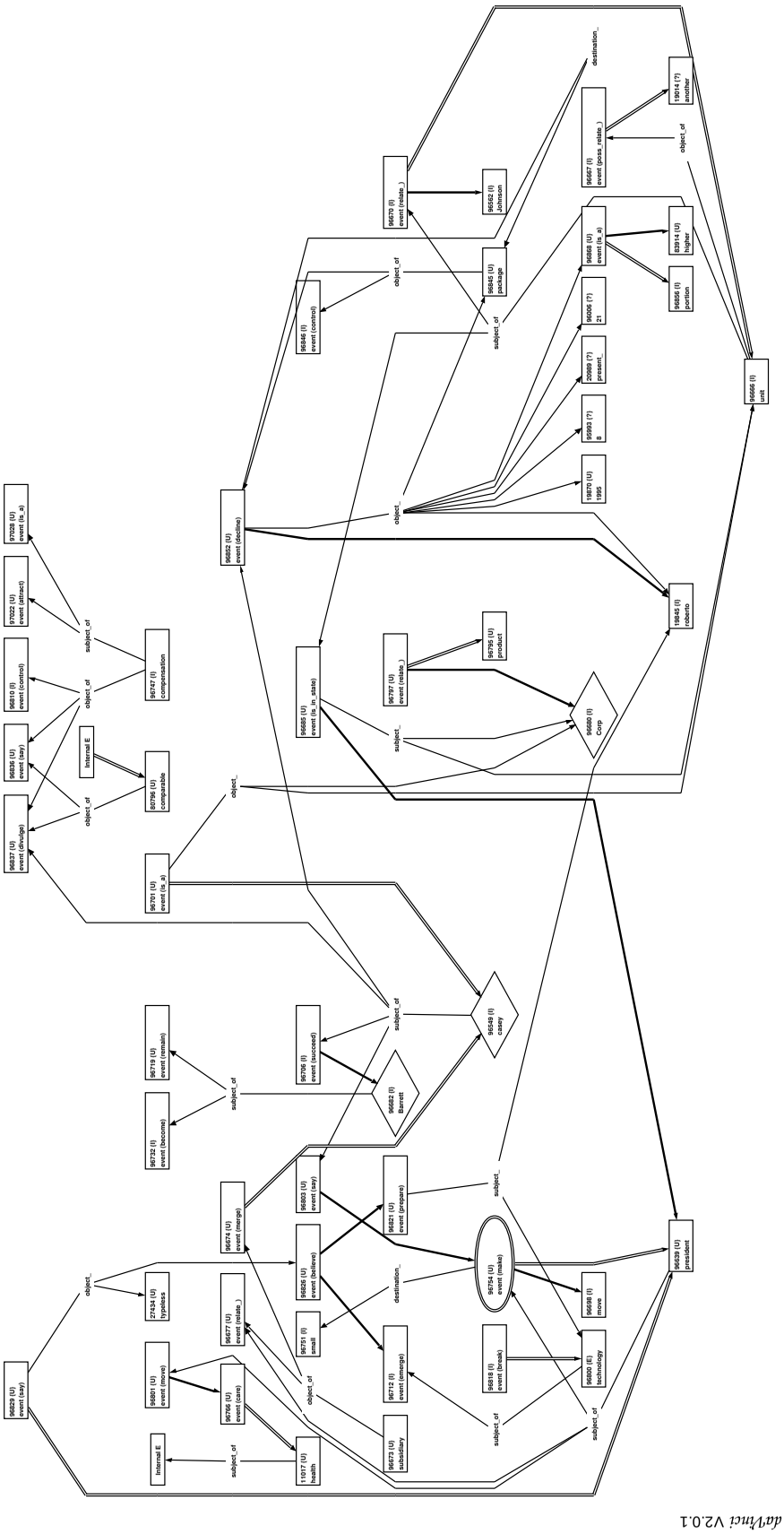


Figure 4.5: Semantic distance result for node 96754; see text for key.

This template only ties the others together, so needs no comment.

- **Succession events 1 and 3**

```
<SUCCESSION_EVENT-1> :=
        a    {SUCCESSION_EVENT-96754}     sc 50.00
   POST:                 president
   VACANCY_REASON:       OTH_UNK *
                       / REASSIGNMENT
   IN_AND_OUT:
        <IN_AND_OUT-1>        --    a {IN_AND_OUT-966820096732}    sc 25.00
        <IN_AND_OUT-2>              a {IN_AND_OUT-965490096754}    sc 50.00
        <no key equiv>        ++    a {IN_AND_OUT-965620096754}
        <IN_AND_OUT-5>        ++    a {IN_AND_OUT-966820096754}    sc 75.00
        <no key equiv>        ++    a {IN_AND_OUT-198450096754}
   SUCCESSION_ORG:
        <ORGANIZATION-1>      --    a {ORGANIZATION-96580}   sc 66.67
        <ORGANIZATION-2>      ++    a {ORGANIZATION-96680}   sc 50.00

<SUCCESSION_EVENT-3> :=
        a    {SUCCESSION_EVENT-96706}     sc 16.67
## POST:                 president
                  XX  a= a top Johnson & Johnson manager
## VACANCY_REASON:       REASSIGNMENT
                  XX  a= OTH_UNK
   IN_AND_OUT:
        <IN_AND_OUT-4>              a {IN_AND_OUT-965490096706}    sc 25.00
        <no key equiv>        ++    a {IN_AND_OUT-966820096706}
        <no key equiv>        ++    a {IN_AND_OUT-198450096706}
        <no key equiv>        ++    a {IN_AND_OUT-965620096706}
        <no key equiv>        ++    a {IN_AND_OUT-198740096706}
   SUCCESSION_ORG:
        <ORGANIZATION-2>      --    a {ORGANIZATION-96680}   sc 50.00
        <ORGANIZATION-1>      ++    a {ORGANIZATION-96580}   sc 66.67
```

These are listed together because it appears that the scorer is incorrect in the mapping. Event 1 should represent the change of president at Genetic Therapy, and event 3 the loss of president at McNeil Pharmaceutical. The matched LOLITA events are (internally) of 'a president makes a move' (the president is not directly identified with a named person), and of 'Casey succeeds Barrett' respectively. From a *semantic* viewpoint, the scorer has mismatched these.

This is unfortunate for diagnostic purposes, as it suggests problems which are not there and hides real problems at the same time. Note that mismatches can also occur in sub-templates, which adds to the difficulty of analysis. For this reason and given the over-generation of IN_AND_OUT templates, our analysis will concentrate on the non-link slots from here on.

The author accepts that template matching of this kind is a difficult task, and suggests that comparisons like that attempted here may not be useful when a system is performing below a certain level of recall and/or precision. For the over-generated templates, quite often several can match one key template equally well, and which one is chosen is essentially arbitrary. This limitation of scoring has the implication that at low scores, the scorer is marking coincidences and mistakes rather than genuine reasoned output. Hence, the marks obtained by weak systems are not on the same scale as much better systems, or, the scale is not linear. Possible ways out of this problem could be by changing the style of templates, as discussed in section 6.2.2, or by weighting the matching of template slots by the strength of sub-matches or by confusibility of fill.

Allowing for the mismatches, the basic LOLITA events are close to the actual events. The correct organisation is identified in SUCCESSION_EVENT-96706, though the post is wrong – the generated one is the closest title to Casey, who succeeds Barrett in that event. The other organisation, which Casey leaves, is incorrectly identified as Ortho Pharmaceutical, though a match is allowed because that organisation's type is correct. This organisation is the nearest to 96754, and McNeil Pharmaceutical is not inside the node pool.

Comparatively, for SUCCESSION_EVENT-1, most systems thought the VACANCY_REASON was REASSIGNMENT, and half of the systems were wrong on IN_AND_OUT-2 – or at least, the scorer did not grant suitable matches. For SUCCESSION_EVENT-3, few systems were correct with the SUCCESSION_ORG, and there was some over-generation for the IN_AND_OUT.

- **Succession event 2**

```
<SUCCESSION_EVENT-2> :=
        a    {SUCCESSION_EVENT-96632}     sc 50.00
## POST:                chief operating officer
                  XX  a= its president
   VACANCY_REASON:      OTH_UNK
   IN_AND_OUT:
        <IN_AND_OUT-3>              a {IN_AND_OUT-965490096632}    sc 50.00
        <no key equiv>        ++    a {IN_AND_OUT-198450096632}
        <no key equiv>        ++    a {IN_AND_OUT-965620096632}
        <no key equiv>        ++    a {IN_AND_OUT-198740096632}
        <no key equiv>        ++    a {IN_AND_OUT-966820096632}
   SUCCESSION_ORG:
        <ORGANIZATION-1>            a {ORGANIZATION-96580}   sc 66.67
```

This template is close. The LOLITA event is of Casey becoming "its president and chief operating officer", although 'its' is erroneously resolved to Casey himself. The basic candidate for the job phrase is "its president and chief operating officer", which is truncated at the 'and' during calculation of the slot fill from the textrefs. The 'its' should not be included – thus the filter on genitives does not catch this form of possessive. It was noted above that this phrase is incorrectly parsed and hence the two job components are not analysed separately. The organisation is picked up as the one Casey was[4] a manager of, which is mistakenly determined to be Genetic Therapy, probably caused by a wrong parse on that sentence: hence, this answer is produced on the basis of the organisation's proximity to the succession event, rather than being directly computed.

Comparatively, two systems miss this template, with nine scoring, three with good marks. Three of the nine get the POST wrong. Six wrongly produce REASSIGNMENT for the VACANCY_REASON. Only three score for the IN_AND_OUT slot, with several systems producing two incorrect fills. For the SUCCESSION_ORG, two systems suggest the wrong ORGANIZATION, and two fail to produce one. The remainder are correct.

- **Succession event 4**

```
<SUCCESSION_EVENT-4> :=
        a    {SUCCESSION_EVENT-96732}     sc 54.55
   POST:                chairman
   VACANCY_REASON:      OTH_UNK
   IN_AND_OUT:
        <IN_AND_OUT-5>        --    a {IN_AND_OUT-966820096754}    sc 75.00
        <no key equiv>        ++    a {IN_AND_OUT-965490096732}
        <no key equiv>        ++    a {IN_AND_OUT-198450096732}
        <no key equiv>        ++    a {IN_AND_OUT-198740096732}
        <IN_AND_OUT-1>        ++    a {IN_AND_OUT-966820096732}    sc 25.00
   SUCCESSION_ORG:
        <ORGANIZATION-1>            a {ORGANIZATION-96580}   sc 66.67
```

The string slots and organisation are correct for this template, which is based on the LOLITA event of Barrett becoming chairman – although chairman of what is not explicitly computed. The title is connected directly to the particular event. 'To become' is neither a

[4] The associated temporal information is both 'present' and 'past' (a bug), the 'past' caused by the past tense of "to move".

departing or reassignment action, hence the vacancy reason is OTH_UNK. The organisation is the closest to the succession event in the node pool.

Only five other systems get this template, and their scores are all much higher than LOLITA's: LOLITA is penalised for over-generation of IN_AND_OUTs.

- **Mapped IN_AND_OUTs for Barrett**

```
<IN_AND_OUT-1> :=
        a    {IN_AND_OUT-966820096732}    sc 25.00
## NEW_STATUS:          OUT
                 XX  a= IN
## ON_THE_JOB:          NO
                 XX  a= UNCLEAR

<IN_AND_OUT-5> :=
        a    {IN_AND_OUT-966820096754}    sc 75.00
   NEW_STATUS:          IN
   ON_THE_JOB:          UNCLEAR
```

For these, the REL_OTHER_ORG and OTHER_ORG slots have been removed. They are not produced in LOLITA responses as they are not yet implemented. Also removed in this section only is the IO_PERSON, which is always Barrett. These templates are produced by virtue of Barrett appearing within the node pool of the respective succession events. For the first, depending on event 96732, NEW_STATUS is determined to be IN, as 'becoming' is set as an IN action in the code, and ON_THE_JOB is UNCLEAR as an event time of 'present' is recorded. For the second, which depends on 96754, making a move does not trigger any of the specific cases, hence the defaults of IN and UNCLEAR. These templates are identical, and hence confusible during scoring since their parents are not taken into account: note the pattern of under-generation and over-generation in SUCCESSION_EVENTs 1 and 4 indicates a swap.

All systems get the first template, with very good marks. Only two systems produce the REL_OTHER_ORG slot, and these two plus another produce the OTHER_ORG slot; all fills are correct. The IO_PERSON is correct for all. For the second template, none score badly. NEW_STATUS is unanimously correct, with around half getting ON_THE_JOB and REL_OTHER_ORG. The three (including LOLITA) who miss the latter do not produce an OTHER_ORG. Four systems get the person wrong, asserting Casey instead of Barrett.

- **Mapped IN_AND_OUTs for Casey**

As before, the unimplemented slots are omitted. The templates are all produced because Casey is within the node pool of the respective events.

```
<IN_AND_OUT-2> :=
        a    {IN_AND_OUT-965490096754}    sc 50.00
   NEW_STATUS:          IN
## ON_THE_JOB:          YES
                 XX  a= UNCLEAR

<IN_AND_OUT-3> :=
        a    {IN_AND_OUT-965490096632}    sc 50.00
   NEW_STATUS:          IN
## ON_THE_JOB:          YES
                 XX  a= UNCLEAR

<IN_AND_OUT-4> :=
        a    {IN_AND_OUT-965490096706}    sc 25.00
## NEW_STATUS:          OUT
                 XX  a= IN
## ON_THE_JOB:          NO
                 XX  a= UNCLEAR
```

The first template's values are produced for the same reason as Barrett's second IN_AND_OUT template. The second relates to Casey becoming president and CEO. The defaults IN and UNCLEAR are produced as the main verb is 'to become', with a time of the present. In the third, the main verb is 'to succeed', which is categorised as a succession action, neither an IN or an OUT action, hence the defaults of IN and UNCLEAR.

Comparatively, LOLITA's score is average for IN_AND_OUT-2. All systems get NEW_STATUS correct, but only three are correct for ON_THE_JOB. *No* system gets REL_OTHER_ORG – systems are equally divided in saying SAME_ORG or omitting the slot, and only two get the OTHER_ORG, with a few suggesting the wrong ORGANISATION. All are correct with IO_PERSON.

LOLITA's score is again average for IN_AND_OUT-3. NEW_STATUS is wrong for only one system, and ON_THE_JOB correct for only one, almost all saying UNCLEAR. Only one system is correct for REL_OTHER_ORG and SAME_ORG, the others split between omission and the wrong answer (an omission (error) in one was often matched by an omission (error) in the other slot). As before, all are correct on the IO_PERSON. All score much more than LOLITA in IN_AND_OUT-4: the main mark-loser is the omission of both REL_OTHER_ORG and SAME_ORG by six systems. Two systems suggest the wrong IO_PERSON.

- **Over-generated IN_AND_OUTs**

```
<MISMATCHED or OVERGENERATED> := a-{IN_AND_OUT-965490096732}
?? NEW_STATUS:          IN
?? ON_THE_JOB:          UNCLEAR
?? IO_PERSON:
        <PERSON-1>          ??    a {PERSON-96549}     sc 100.00

<MISMATCHED or OVERGENERATED> := a-{IN_AND_OUT-965620096632}
?? NEW_STATUS:          IN
?? ON_THE_JOB:          UNCLEAR
?? IO_PERSON:
        <no key equiv>      ??    a {PERSON-96562}

<MISMATCHED or OVERGENERATED> := a-{IN_AND_OUT-966820096632}
?? NEW_STATUS:          IN
?? ON_THE_JOB:          UNCLEAR
?? IO_PERSON:
        <PERSON-2>          ??    a {PERSON-96682}     sc 66.67

<MISMATCHED or OVERGENERATED> := a-{IN_AND_OUT-966820096706}
?? NEW_STATUS:          IN
?? ON_THE_JOB:          UNCLEAR
?? IO_PERSON:
        <PERSON-2>          ??    a {PERSON-96682}     sc 66.67

<MISMATCHED or OVERGENERATED> := a-{IN_AND_OUT-965620096706}
?? NEW_STATUS:          IN
?? ON_THE_JOB:          UNCLEAR
?? IO_PERSON:
        <no key equiv>      ??    a {PERSON-96562}

<MISMATCHED or OVERGENERATED> := a-{IN_AND_OUT-965620096754}
?? NEW_STATUS:          IN
?? ON_THE_JOB:          UNCLEAR
?? IO_PERSON:
        <no key equiv>      ??    a {PERSON-96562}
```

These are all produced because the IO_PERSON in them appeared within the respective succession event's node pool. The NEW_STATUS and ON_THE_JOB were determined as for the other IN_AND_OUTS of the respective succession event. As noted above, there is no test for a person's relevance to an event, hence the over-generation. No other system produced more than three over-generated IN_AND_OUT templates.

- **Erroneous IN_AND_OUTs**

```
<MISMATCHED or OVERGENERATED> := a-{IN_AND_OUT-198450096632}
?? NEW_STATUS:          IN
?? ON_THE_JOB:          UNCLEAR

<MISMATCHED or OVERGENERATED> := a-{IN_AND_OUT-198450096706}
?? NEW_STATUS:          IN
?? ON_THE_JOB:          UNCLEAR

<MISMATCHED or OVERGENERATED> := a-{IN_AND_OUT-198450096732}
?? NEW_STATUS:          IN
?? ON_THE_JOB:          UNCLEAR

<MISMATCHED or OVERGENERATED> := a-{IN_AND_OUT-198450096754}
?? NEW_STATUS:          IN
?? ON_THE_JOB:          UNCLEAR

<MISMATCHED or OVERGENERATED> := a-{IN_AND_OUT-198740096632}
?? NEW_STATUS:          IN
?? ON_THE_JOB:          UNCLEAR

<MISMATCHED or OVERGENERATED> := a-{IN_AND_OUT-198740096706}
?? NEW_STATUS:          IN
?? ON_THE_JOB:          UNCLEAR

<MISMATCHED or OVERGENERATED> := a-{IN_AND_OUT-198740096732}
?? NEW_STATUS:          IN
?? ON_THE_JOB:          UNCLEAR
```

These involve the concept of either LOLITA itself (19874) or R. Garigliano (19845). Note that no IO_PERSON is produced: all person templates must contain PER_NAME, which is produced from the textrefs attached to the concept. As the names of these concepts do not appear in the article, no suitable PER_NAME is found. Those concepts appear relevant to the article analysis because of the weak handling of reported speech. For example, 'I' in reported speech is not resolved to the speaker, but to LOLITA's default source of information, R. Garigliano.

- **The people**

```
<PERSON-1> :=
        a    {PERSON-96549}     sc 100.00
   PER_ALIAS:           Casey
   PER_NAME:            Michael D. Casey
   PER_TITLE:           Mr.

<PERSON-2> :=
        a    {PERSON-96682}     sc 66.67
   PER_ALIAS:           Barrett
## PER_NAME:           M. James Barrett
                XX  a= James Barrett
   PER_TITLE:           Mr.

<MISMATCHED or OVERGENERATED> := a-{PERSON-96562}
?? PER_ALIAS:           Genetic Therapy : J&J
?? PER_NAME:            Johnson & Johnson
```

These two matched templates and the unmatched one are syntactically identical to those from the TE task. The J&J 'person' is not relevant to the changeover, so it is an over-generation. In context, it means the person coming IN to the 'jobs' of "its president" and "a top Johnson & Johnson manager" for Genetic Therapy, and of 'president' of Ortho Pharmaceutical, for all of which ON_THE_JOB is UNCLEAR. It is generated because the concept is within the respective succession events' node pools.

All systems score perfect marks on the Casey template, and highly on the Barrett template (LOLITA's score is lowest, as the PER_NAME is incorrect). One system produces a template for Hawkins.

- **The organisations**

```
<ORGANIZATION-1> :=
        a    {ORGANIZATION-96580}   sc 66.67
   ORG_ALIAS:           Genetic Therapy
## ORG_COUNTRY:         United States
                  --     a
## ORG_DESCRIPTOR:      a small biotechnology concern here
                       / a small biotechnology concern
                       / the smaller company
                  --     a
## ORG_LOCALE:          GAITHERSBURG CITY
                  --     a
   ORG_NAME:            Genetic Therapy Inc.
   ORG_TYPE:            COMPANY
```

LOLITA's template for the company in which the main changes occur is identical to the TE output, as is the f-score. However, there is a difference between the key version of this template for ST and TE: in ST, the indefinite phrase "a small biotechnology concern" is permitted, whereas 'here' was required in the TE key. Since the task definition for the content of ORGANIZATIONs and PERSONs is identical, we would expect the keys to be the same. The issue of key consistency is considered in section 6.2.3.

Comparatively, LOLITA's score is average, with two systems not producing anything. No system produces the country or locale slots. The descriptor is the next biggest problem, only three systems scoring.

```
<ORGANIZATION-2> :=
        a    {ORGANIZATION-96680}   sc 50.00
## ORG_NAME:            McNeil Pharmaceutical
                  XX  a= Ortho Pharmaceutical Corp.
   ORG_TYPE:            COMPANY
```

This template should be for the company that Casey leaves, but indicates that Casey leaves Ortho Pharmaceutical Corp. The type is determined by the suffix of the erroneous name. It is produced as the nearest organisation in the node pool to the succession event 96754.

Only *one* system is correct in this template. One other produces the same name as LOLITA, and another produces 'Johnson & Johnson'. So, only four systems score, and three of these effectively score only because they suggest something of type company which is not "Genetic Therapy". Note the inconsistency between this and the TE version – the "ORG_DESCRIPTOR: subsidiary" is missing from this template. No matching system generated a descriptor.

4.6 Current Performance on this Article

After approximately one person-year's worth of improvement work, LOLITA was rerun on the specimen article. The first section outlines the changes made, then the effects on each of the tasks is presented. The template comparison tool was very useful in assessing the qualitative changes.

4.6.1 The Changes to LOLITA

Most of this work was done by R. Garigliano, when time allowed, and typically involved attempting to raise scores on a small test set from the training articles. As scores improved, the test set was expanded[5]. Additionally, work on other projects aided the MUC-6 performance by improving the core, such as work by R. Garigliano and R. Morgan on other template-based tasks. Very little work was done on the actual task implementations. The author did not contribute significantly to this post-MUC-6 work.

- Much more use is made of the "pre-parser" – it now recognises and packages many candidate Named Entities before conventional parsing. The result is that NE handling, especially names, is much improved.
- The feature system in the grammar is more expressive, leading to a much more compact grammar, with some improvements in maintainability.
- The Semantic Net data was improved. There were a number of logical problems related to our use of the WordNet data in circumstances not envisaged by its designers. Tasks included checking the consistency of basic information in hierarchies, and integration with existing portions of the net.
- Much routine system-wide debugging work was performed.
- Task-wise, little work was done apart from fixing bugs in the implementations. Some work was done between the actual evaluation and the MUC-6 conference to improve performance on the walk-through article (which was discussed by all participants), such as extending the concepts relevant to the succession scenarios.

Of this work, only the pre-parser was significantly driven by MUC needs. The remainder is general system work.

4.6.2 Named Entity

The scores are now 100% for both recall and precision. The biggest source of error was mis-typing a company because part of the name was a common surname, which would have shown up in many articles. This class of problem is now clearly corrected.

4.6.3 Coreference

Recall increases from 18/37 to 22/37, and precision from 18/32 to 22/29, giving new percentages of 59% and 76%. Four additional links have been gained, and three over-generated links removed. From the key to response mapping graph (as in figure 4.1), it is clear that chains are more independent, resulting in less intersection of chains. Quite often, instead of the offending link now being part of the correct chain, it is not produced. The comparison tool output for the original and the new version were compared: with the fixed output format of the tool, something as simple as `diff` can be used. The chain-specific scores helped significantly in determining the change in performance. We now consider each chain in turn, and use the `K_n` notation introduced in section 4.3. Optionals are included again, due to the problem with the scorer.

The score in `K_0` is unchanged, with the 'chairman' markup now omitted. `K_1` no longer scores, losing a point of recall and precision: one of the scoring markups now loses precision in another chain. The response chain mapping to `K_2` gains an extra markup, losing precision. `K_3` is no longer produced by LOLITA, the same for `K_4` and `K_5`. The response markups originally mapping to parts of these chains no longer appear, improving precision overall.

[5]Most of this work was done on the following 15 articles from the formal training set: 940124-0017, 940407-0168, 930209-0147, 940324-0097, 931021-0152, 940114-0038, 940425-0043, 930402-0074, 940111-0056, 930319-0065, 940310-0087, 940325-0108, 930927-0024, 940302-0061, 930412-0090.

The biggest change is seen in K_6, where the pronouns for Casey are linked to the correct chain. Formerly, this key chain was mapped to three response chains. Two of these are merged to produce one big chain, with a score line of `sc R:9, P:9 => r: 11/16 p: 9/11`. Overall, recall for K_6 is increased by 3 points, and precision is much improved. The occurrence of Casey which is ignored by the scorer is still ignored, and there is no great change in the number of missed markups in this chain.

K_7 achieves full recall by supplying the missing markup, but also over-generates by one point. K_8 gains two of its missing markups, but doesn't over-generate, hence plus two for recall while precision remains perfect. K_9 is not produced, as before. There are many fewer unmatched templates.

So, the output is more conservative, losing recall in a few chains but gaining precision overall. Recall increases slightly as the gains exceed the losses. The most significant changes appear to be the reduction of over-generation, and the better resolution of the pronouns in this article. It is hard to suggest where more work should be directed, but attention is needed for the links which are now lost.

4.6.4 Template Elements

Recall and precision are 64% and 93%, compared to the previous 55% and 75%. The most significant change is related to identifying 'J&J' as a company, and getting that template perfectly correct. The 'Genetic Therapy' from the title is no longer erroneously added to this template. Also appearing is the person template for Hawkins, which gets full marks.

However, the complete template for Barrett is lost. The reason appears to be that 'M. James Barrett' is categorised as a normal noun phrase, whereas the PER_NAME must be a proper noun. Person templates must contain a PER_NAME. This bug could be a consequence of the changes in proper name handling, since the previous version had this template. However, this template does appear in the ST output! The basic implementation of PERSON templates is common to both tasks, so the reason for the difference must be at a higher level.

The other mistake is in the ORG_DESCRIPTOR for 'Genetic Therapy'. Interestingly, by the ST key, it would be correct (see the comments about key consistency above). This error is the reason for the precision drop. The country and locale for this organisation are again missed, as are the ORG_DESCRIPTORs for the other two organisations.

The output for this task is too small to make general statements about the improvement. The bug causing loss of 'Barrett' is significant, and the ORG_DESCRIPTOR calculation still needs work. The author suspects that the main problem lies in parsing the expressions, as suitable textref sequences (ie, grouped by virtue of being in a correct subtree, see section 3.2.5) are not being attached during semantics.

4.6.5 Scenario Templates

From recall 53% and precision 34% to 52% and 67% indicates much better precision with little change in recall, and supports the view that the current LOLITA is more conservative. However, this is slightly illusory for the template task. Firstly, one source of imprecision, the templates involving LOLITA and R. Garigliano have been removed by a simple filter. Secondly, SUCCESSION_EVENT-3 has no equivalent in the response, which reduces recall but does not further reduce precision. Thirdly, 'Johnson & Johnson' is now correctly analysed as a company, hence one less 'person' on which to build over-generated templates. The underlying problem of insufficient checks of relevance for a sub-template remains.

With the loss of SUCCESSION_EVENT-3, SUCCESSION_EVENT-1 is correctly mapped by the scorer. Its score increases because the SUCCESSION_ORG is as expected, and some of the over-generated IN_AND_OUTs are lost. However, no IN_AND_OUT is correct – two are added, two are omitted. SUCCESSION_EVENT-2 drops its scoring IN_AND_OUT link and omits an over-generation, so a slight drop in f-score. SUCCESSION_EVENT-4 now contains the expected

IN_AND_OUT link, and has reduced over-generation, hence a score increase.

There is no change in content or score for the five expected IN_AND_OUT links and, as noted above, the over-generation has been greatly reduced due to a simple filter, the non-production of the incorrect person, and loss of a SUCCESSION_EVENT. The ORG_DESCRIPTOR in ORGANIZATION-1 scores here, whereas it didn't in TE because of an omission in the TE keys. ORGANIZATION-2 is missing from the response; it is the organisation in the omitted SUCCESSION_EVENT-3. The PERSON templates are unchanged. There is currently only one unmatched template.

4.7 Conclusions

The conclusions on task performance are necessarily anecdotal, given the small collection of evidence.

4.7.1 Named Entity

The errors are few (six), but are traceable to just two analysis errors. These are manifestations of one basic error: the failure to merge conflicting semantic information in particular cases. Note that the six errors did not occur with the current LOLITA: this was a fundamental bug which was noticed for many articles, and was easily corrected.

4.7.2 Coreference

To summarise, LOLITA gets most of the straightforward features, but does make mistakes in simple cases. The multi-system analysis shows that there is little that LOLITA gets correct that other systems do not. We could attempt to classify the errors, but will not. A preliminary analysis shows that there are no 'big' causes of errors which can be easily corrected. The errors occur at all stages of analysis (missing data, semantics errors, task failures, &c). In a task as complex as Coref, some task errors can be caused by a series of core analysis errors. Thus, we cannot identify particular areas of the system which need urgent improvement.

For the multi-system comparison, even with the small number of other systems, it is not as easy as one would wish to extract useful information: for example, concerning when LOLITA is doing better than other systems or significantly worse. Part of this is due to the complexity of coreference. One possible strategy could be comparison against the highest scoring other system in each chain.

4.7.3 Template Element

The NE type errors lead to the loss of two templates, and significant loss of points. Smaller errors are in poor descriptor production and misunderstanding abbreviated first names. Two slots are unimplemented. Ignoring the lost templates, LOLITA performs relatively well against other systems, though is never top in any template. A moderate score seems easy to obtain, using simple techniques.

4.7.4 Scenario Template

As with Coref, the task complexity makes analysis hard. LOLITA's main losses are through over-generation, unimplemented slots, and problems with alignment in scoring. The over-generations were caused by a simple (and now corrected) bug, plus the underlying approach of using semantic distance, which is necessitated by weaknesses of the core analysis. Some problems could be alleviated with some test of relevance between a main event node and candidate sub-templates. The scoring problems occur when a system produces similarly-scoring templates that have different

parents. Mismapping can occur, resulting in the relevant slot in both parents being marked as incorrect. Other systems appear to suffer from this problem in this article.

Comparatively, LOLITA's recall is just below average, but with very poor precision. The losses of other systems are through scorer misalignment, failing to produce templates, and mixed errors on all slots. In particular, ORG_LOCALE and ORG_COUNTRY is omitted by all systems, and OTHER_ORG and REL_OTHER_ORG are poorly filled by all.

4.7.5 Current Performance

There are small improvements overall. The best score changes are in precision, but a lot of this can be attributed to correction of bugs which formerly caused over-generation. Considering the results qualitatively, with the template comparison tool, there are several cases of marks in one area being exchanged for marks in another. We conclude that LOLITA is being more conservative: aggressive heuristics have been tightened up to improve precision, at the cost of losing some recall.

4.7.6 Overall View

LOLITA performance on all four tasks has been analysed in detail for one article (chosen to be representative), and has also been compared against the performance of other systems on that article. There is some evidence that LOLITA is able to score on the more complex parts of a task, but is held back by making relatively simple mistakes. LOLITA is sometimes in the correct minority for features that, universally, are not well done; but it often makes mistakes on features that most systems do get. More evidence is presented in the next chapter (section 5.6 and section 5.7).

Comparison with current LOLITA performance, after one person-year of work, shows a modest improvement. It is not as large as hoped for, but there is evidence of the system becoming more conservative: precision is increased at the cost of lowering recall.

The processing requirements of the MUC-6 tasks have also been indirectly examined for one article; we do not know of such an analysis in the literature. It appears that fairly good marks can be obtained using straightforward techniques and heuristics. During the analysis, several problems with the style of definition and method for scoring the MUC-6 tasks were identified. These are relevant to designs of future evaluations, and are discussed further in chapter 6.

More conclusions are made in the next chapter, after examination of the MUC-6 performance through the scores.

Chapter 5

LOLITA Performance on MUC-6: The Overall Scores

First, we discuss how the scores will be examined, and outline a methodology. Next, the October 1995 MUC-6 scores for LOLITA are examined, for each task. We then consider two different ways of looking at MUC-6 scores which make use of the output of other systems to gain a relative picture of LOLITA's performance. These methods were developed by the author. After this, the current performance of LOLITA is discussed, the result of approximately one person-year's worth of work by various people. We briefly discuss the success of LOLITA's internal analysis on the evaluation texts, then draw conclusions on LOLITA's complete performance in MUC-6. Some familiarity with the MUC-6 tasks is assumed. More details can be found in [DARPA, 1995].

5.1 Examining the Scores

The scores are the main external indicator of performance and the official result of the evaluation of LOLITA in isolation. How are they to be analysed? Apart from making observations of the form "LOLITA scores X on feature Y", it is not obvious. LOLITA was not designed specifically for MUC or MUC-like tasks, so these scores are not a *natural* way of looking at LOLITA. Hence, score interpretation is not straightforward for us. It helps if one has a clear idea of what one wants from the information. The author's interest is mainly diagnostic – seeing where LOLITA is performing well, and when it is deficient – so the remainder of this section will have this bias.

One consequence of the bias is that the scorer output is not of direct diagnostic use: it does not mean much that we only get X% of all occurrences of a specific feature. It is very interesting if X is very high or very low, especially if other LOLITA scores do not show such polarity. But in general, most statistics will be somewhere in the middle band, and the interpretation will be that they are lower than wished. Performance in that statistic normally can't be modified by a local change to the system – for example there is no section of code which handles that feature alone. This means that the scorer output is not directly linkable to system features. This is further complicated in LOLITA as results are derived from a detailed analysis – hence such associations are even harder to make. Additionally, some answers are easier to get than others, so we are more interested in knowing *how* the scores were achieved – a qualitative interest.

For these reasons, few comments will be made on the initial scores summaries. Analysis becomes more interesting when two sets of scores from the same system are compared: one would assume that later scores are improvements so we can look for various features in the data, such as big increases or unexpected drops in some category. Comparing data from different systems is not as easy and is open ended, as we cannot make a priori assumptions about relationships between their performance. Clearly, if one system scores much higher than another for most features, this is significant. A mixture of higher and lower is harder to interpret.

Other viewpoints are possible. One would look for a system scoring above a certain threshold

on certain features before using it for a specific application. Another could look for one system doing better than its competitors, with special attention for the areas where that system is not better.

5.1.1 Analysis Methodology

We present the score summary tables for each task, and comment on interesting features. The score distributions for individual articles are summarised by scatter plots. We also investigate correlations in the score tables.

Format of Tables

For the tables, the column headings are[1] POSsible number in key, ACTual in response; then the numbers CORrect, PARtially correct (this isn't used in MUC-6), INCorrect, SPUrious (ie, added), MISsing, and NONcommittal. The last group are ratios: RECall, PREcision, UNDergeneration, OVer-Generation, SUBstitution, and ERRor per response fill. These are fully explained in [DARPA, 1995], p 293. Probably the most important numbers are the RECall and PREcision entries in the ALL OBJECTS line. The f-measures are combinations of these two numbers.

Explanation of Graphs

An utility to produce graphs of any two columns from the score reports was written by the author, using `gnuplot` to draw the final output. It can also plot particular slot results from the reports, eg for PER_NAME. Points in the graphs are labelled with the size rank of the article giving that score (higher ranks mean larger articles), to help investigate the relationship between scores and article size. Appendix D contains the article to size rank mapping used.

The size is defined as the raw token count, so does include the SGML tags. This is not a concern as all articles have a fairly uniform size 'prelude' to the main text, and a small number of paragraph markers roughly proportional to the text body size. The rank is taken over all 100 articles; so the NE-CO subset is within this, hence the disjoint numbers. Quartile lines for each variable are also shown, to help judge the skewness of the distribution: a quarter of the data lie between each pair of lines.

Statistical Analysis: Correlation

Since no assumptions about the shape of score distributions can be defended[2], non-parametric methods of statistical analysis are required. [Sprent, 1989] explains the relevant techniques. The commonly-used Spearman's coefficient of rank correlation is one suitable measure. Several combinations of variables are examined: eg, article size rank, REC, PRE, original and current scores, scores from certain template slots. We note only the interesting ones – especially ones that go against (intuitive) expectations. For example, we wonder if article size and recall correlate[3]. Alternatively, REC and COR obviously correlate as the former depends on the latter.

The calculation method used is taken from [Sprent, 1989], page 136. The calculations were done with code written by the thesis author, with some results verified using a subroutine from a standard mathematical library. Table A9 in Sprent is used for the critical absolute values of the coefficient. Unless otherwise stated, the confidence level is 5% in the statements of significance.

[1] Please note: the different versions of the scorers are not consistent in column order. The main difference is in the order of the ratios.

[2] "We do not know the distribution of our sample and prefer not to make an assumption about it. Computationally-intensive methods empirically generate the distribution." [Chinchor, 1993]

[3] Note that correlation does not indicate causality: larger articles could include longer sentences, which are a known problem for LOLITA, hence a loss in recall.

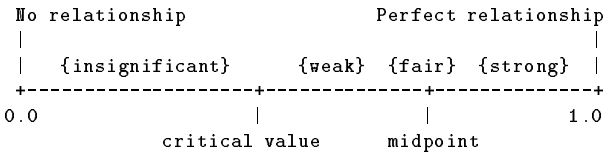


Figure 5.1: Interpretation of correlation coefficient.

There are no hard rules for the interpretation of fractional magnitudes of the coefficient. Choices are usually dictated by the application: physical scientists require very high fractions for a good relationship, whereas social scientists are content with less. In this work, we adopt the rough scale in figure 5.1. Anything below the critical value (CV) will be interpreted as "insignificant relationship", anything up to half-way between the critical value and one as "weak relationship", numbers around the midpoint between CV and 1.0 as "fair relationship", and the upper section as "strong relationship". Zero represents lack of relationship, and one a perfect relationship. Negative coefficients indicate negative relationships: a decrease in one variable linking to an increase in the other.

5.2 Named Entity

SLOT	POS	ACT	COR	PAR	INC	SPU	MIS	NON	REC	PRE	UND	OVG	ERR	SUB
<enamex>	937	772	649	0	0	123	288	0	69	84	31	16	39	0
type	937	772	525	0	124	123	288	0	56	68	31	16	50	19
text	937	772	580	0	69	123	288	0	62	75	31	16	45	11
subtotals	1874	1544	1105	0	193	246	576	0	59	72	31	16	48	15
<timex>	111	101	85	0	0	16	26	0	76	84	23	16	33	0
type	111	101	85	0	0	16	26	0	76	84	23	16	33	0
text	111	101	74	0	11	16	26	0	67	73	23	16	42	13
subtotals	222	202	159	0	11	32	52	0	72	79	23	16	37	6
<numex>	93	74	74	0	0	0	19	0	80	100	20	0	20	0
type	93	74	74	0	0	0	19	0	80	100	20	0	20	0
text	93	74	74	0	0	0	19	0	80	100	20	0	20	0
subtotals	186	148	148	0	0	0	38	0	80	100	20	0	20	0
ALL OBJECTS	2282	1894	1412	0	204	278	666	0	62	74	29	15	45	13
MATCHED ONLY	1616	1616	1412	0	204	0	0	0	87	87	0	0	13	13

	P&R	2P&R	P&2R
F-MEASURES	67.62	71.62	64.05

* * * TASK SUBCATEGORIZATION SCORES * * *

SLOT	POS	ACT	COR	PAR	INC	SPU	MIS	NON	REC	PRE	UND	OVG	ERR	SUB
Enamex:														
organization	453	395	219	0	84	92	150	0	48	55	33	23	60	28
person	373	285	248	0	21	16	104	0	66	87	28	6	36	8
location	111	92	58	0	19	15	34	0	52	63	31	16	54	25
Timex:														
date	111	101	85	0	0	16	26	0	76	84	23	16	33	0
Numex:														
money	76	61	61	0	0	0	15	0	80	100	20	0	20	0
percent	17	13	13	0	0	0	4	0	76	100	24	0	24	0

* * * DOCUMENT SECTION SCORES * * *

SLOT	POS	ACT	COR	PAR	INC	SPU	MIS	NON	REC	PRE	UND	OVG	ERR	SUB
HL	136	80	62	0	12	6	62	0	46	78	46	8	56	16
DD	60	60	60	0	0	0	0	0	100	100	0	0	0	0
DATELINE	52	52	48	0	0	4	4	0	92	92	8	8	14	0
TXT	2034	1702	1242	0	192	268	600	0	61	73	29	16	46	13

Every named entity value must have a type, so one obvious remark is on the consistency: in ENAMEX, more names are correct than types, indicating that the types are wrong in many cases. Notice that the table does not indicate how many times *both* text and type have been correct. The reverse happens for TIMEX, but not for NUMEX: the TIMEX fault could be in marking only part of the expression. It is clear that over-generation is a problem for ENAMEX, as well as the high number of missing markups. From the sub-categorisation table, it appears that organisations are the greatest problem. Locations also score badly, but there are far fewer, hence a lesser effect on scores. Precision is relatively good for person markups. Lastly, the document section scores confirm that title analysis is weak, whereas DD and DATELINE analysis is much better. The recall and precision scatter plot is in figure 5.2.

We now consider correlations in the data. Over the 30 articles in the evaluation, recall and precision are fairly related in named entity. Fair *negative* correlations are seen for both recall and precision against article size (by word count): this indicates that larger articles do show some decrease in recall and precision. One possible reason for this is the typical use of names: often, a full name is introduced, followed by many occurrences of the surname alone. A mistake in the full name could result in wrong type for many occurrences of the surname.

5.3 Coreference

```
Coreference Totals:     Recall: 533/1493 =   0.36
                     Precision: 533/1205 =   0.44
```

Unfortunately, coref scorer output does not contain the rich detail of the other tasks, so little comment can be made about the above. The recall-precision plot is in figure 5.3. Recall and precision show fair correlation. Compared against NE, the two tasks' precision shows no significant correlation, with little correlation for recall. This supports the idea that NE performance does not aid Coref performance. Size-wise, recall and precision are uninfluenced by article size.

5.4 Template Element

SLOT	POS	ACT	COR	PAR	INC	MIS	SPU	NON	REC	PRE	UND	OVG	ERR	SUB
organization	606	622	468	0	38	100	116	0	77	75	17	19	35	8
name	546	622	240	0	229	77	153	0	44	39	14	25	66	49
alias	172	125	62	0	6	104	57	334	36	50	60	46	73	9
descriptor	235	24	8	0	5	222	11	286	3	33	94	46	97	38
type	606	622	453	0	53	100	116	0	75	73	17	19	37	10
locale	114	10	6	0	1	107	3	231	5	60	94	30	95	14
country	115	10	6	0	1	108	3	230	5	60	94	30	95	14
person	496	420	292	0	85	119	43	0	59	70	24	10	46	23
name	496	420	259	0	118	119	43	0	52	62	24	10	52	31
alias	170	146	109	0	0	61	37	190	64	75	36	25	47	0
title	166	152	135	0	0	31	17	185	81	89	19	11	26	0
ALL OBJECTS	2620	2131	1278	0	413	929	440	1456	49	60	35	21	58	24

	P&R	2P&R	P&2R
F-MEASURES	53.80	57.34	50.67

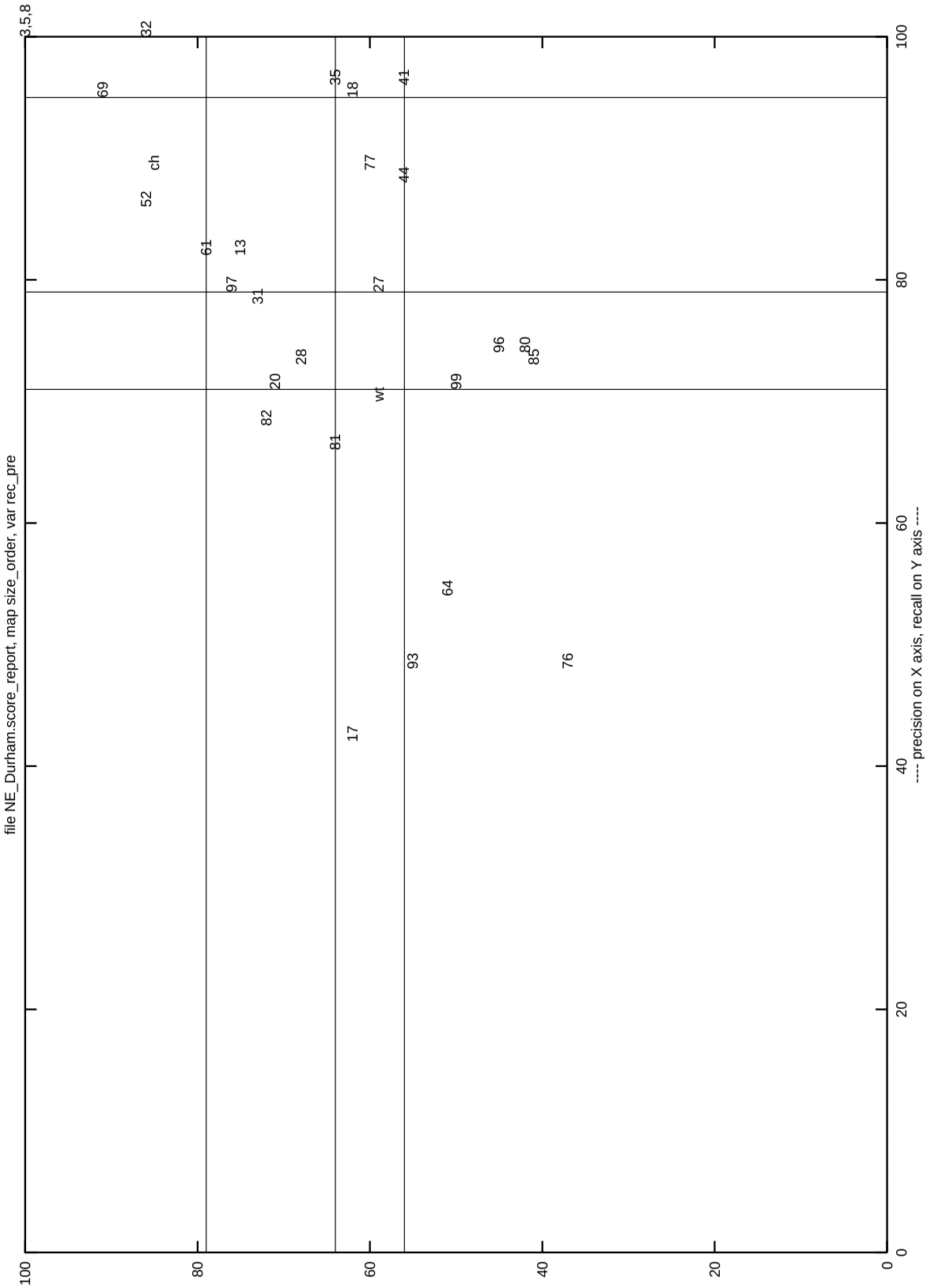


Figure 5.2: Scatter plot of recall (y) and precision (x) for Named Entity. (see appendix D for key.)

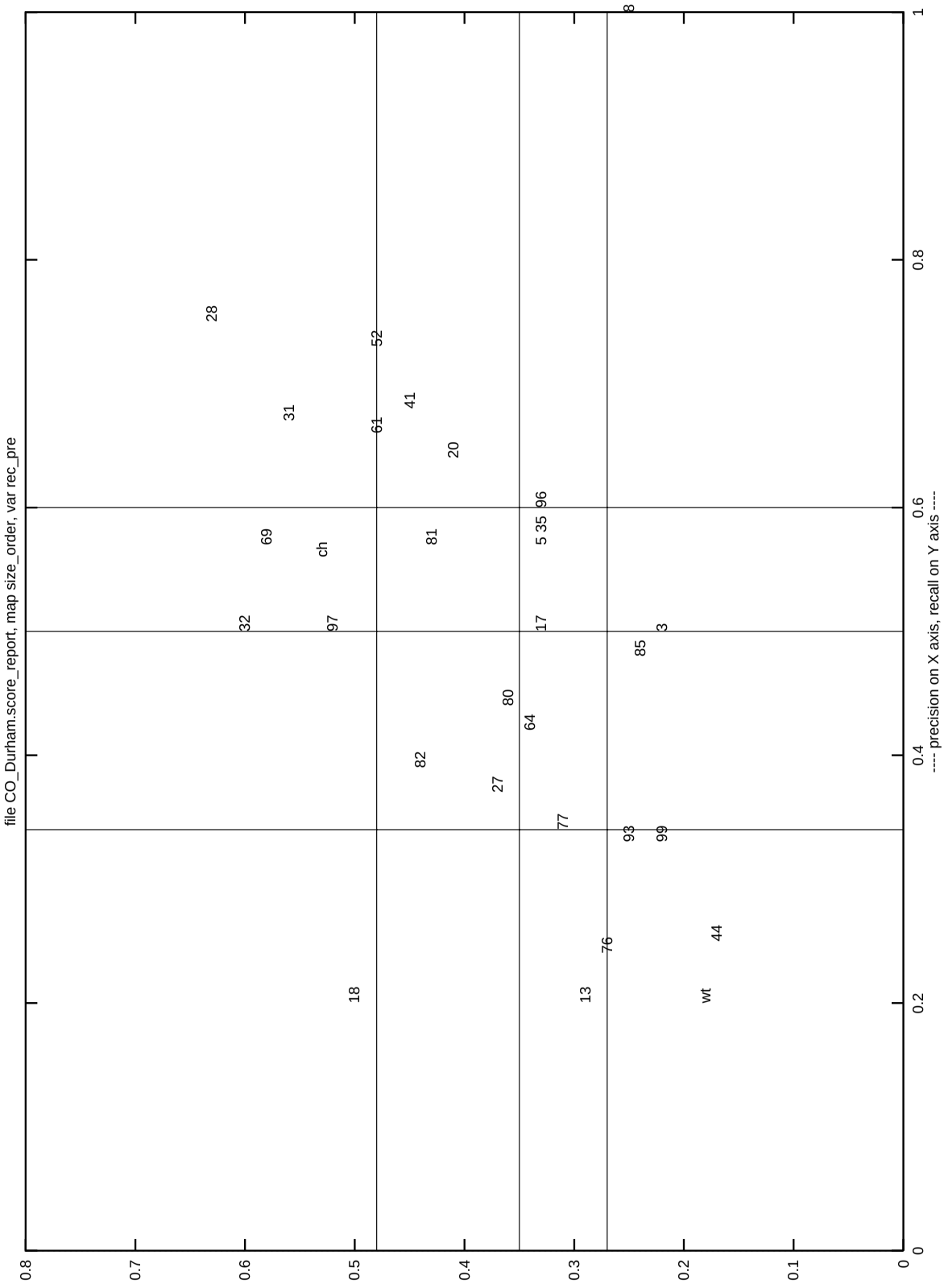


Figure 5.3: Scatter plot of recall (y) and precision (x) for Coreference

Person and organisation names are poorly done. The high number of these mean that the effects on recall and precision are severe. Of particular concern is the high number of incorrect and spurious organisation names, and the very poor performance on descriptors. However, the best score from other systems on the latter is only 38% REC and 51% PRE. Correlation-wise, recall and precision are fairly related, as are both recall and precision for TE against the same for NE on the thirty articles scored in that task. With the similar test for CO, recall is weakly related and precision unrelated. Article size does not influence recall or precision. Figure 5.4 shows the scatter plot for TE.

5.5 Scenario Template

LOLITA produces just over half of the expected succession events, over-generating by around 25%. The templates for these events are poorly done, especially the POST slots. There is much over-generation with the IN_AND_OUT templates, and the slot fills are incorrect for most of the occurrences. OTHER_ORG and REL_OTHER_ORG were not implemented, hence the 100% ERRor. Organisations are also weak – though it is interesting that organisation types are much better than the remaining slots. Descriptors, locales and countries are particularly weak – although there are a smaller number of them. Person templates score slightly better.

SLOT	POS	ACT	COR	PAR	INC	MIS	SPU	NON	REC	PRE	UND	OVG	ERR	SUB
template	53	72	46	0	0	7	26	21	87	64	13	36	42	0
content	181	141	101	0	3	77	37	0	56	72	43	26	54	3
succession_e	191	205	111	0	3	77	91	0	58	54	40	44	61	3
success_or	191	140	57	0	22	112	61	0	30	41	59	44	77	28
post	191	205	29	0	85	77	91	0	15	14	40	44	90	75
in_and_out	256	404	94	0	34	128	276	0	37	23	50	68	82	27
vac_reason	191	205	67	0	47	77	91	0	35	33	40	44	76	41
in_and_out	260	404	162	0	2	96	240	0	62	40	37	59	68	1
io_person	260	331	116	0	30	114	185	0	45	35	44	56	74	21
new_status	260	404	115	0	49	96	240	0	44	28	37	59	77	30
on_the_job	260	404	86	0	78	96	240	0	33	21	37	59	83	48
other_org	175	0	0	0	0	175	0	47	0	0	100	0	100	0
rel_oth_or	175	0	0	0	0	175	0	47	0	0	100	0	100	0
organization	112	93	53	0	3	56	37	0	47	57	50	40	64	5
name	109	93	29	0	26	54	38	0	27	31	50	41	80	47
alias	66	42	22	0	0	44	20	13	33	52	67	48	74	0
descriptor	65	7	2	0	2	61	3	18	3	29	94	43	97	50
type	112	93	53	0	3	56	37	0	47	57	50	40	64	5
locale	41	3	0	0	3	38	0	13	0	0	93	0	100	100
country	41	3	2	0	1	38	0	13	5	67	93	0	95	33
person	133	169	83	0	10	40	76	0	62	49	30	45	60	11
name	133	169	68	0	25	40	76	0	51	40	30	45	67	27
alias	86	86	52	0	1	33	33	23	60	60	38	38	56	2
title	82	89	59	0	0	23	30	21	72	66	28	34	47	0
ALL OBJECTS	2875	2819	952	0	409	1514	1458	195	33	34	53	52	78	30
TEXT FILTER	53	72	46	0	0	7	26	21	87	64	13	36	42	0

F-MEASURES	P&R	2P&R	P&2R
	33.44	33.64	33.24

Not immediately obvious from the table is the effect of false positives in relevancy: it is clear that succession event templates are erroneously produced for 26 articles, but the information about the sub-templates of this over-generation is lost amongst the figures for the relevant articles. We estimate that 10 points of precision are lost in this way. Note that this over-generation cannot be represented in the scatter plot of recall against precision (figure 5.5) – irrelevant articles score zero recall and zero precision as they contain no correct fills (this is set by the scorer: recall will be undefined if calculated as COR over POS, which are both zero). The TEXT FILTER line

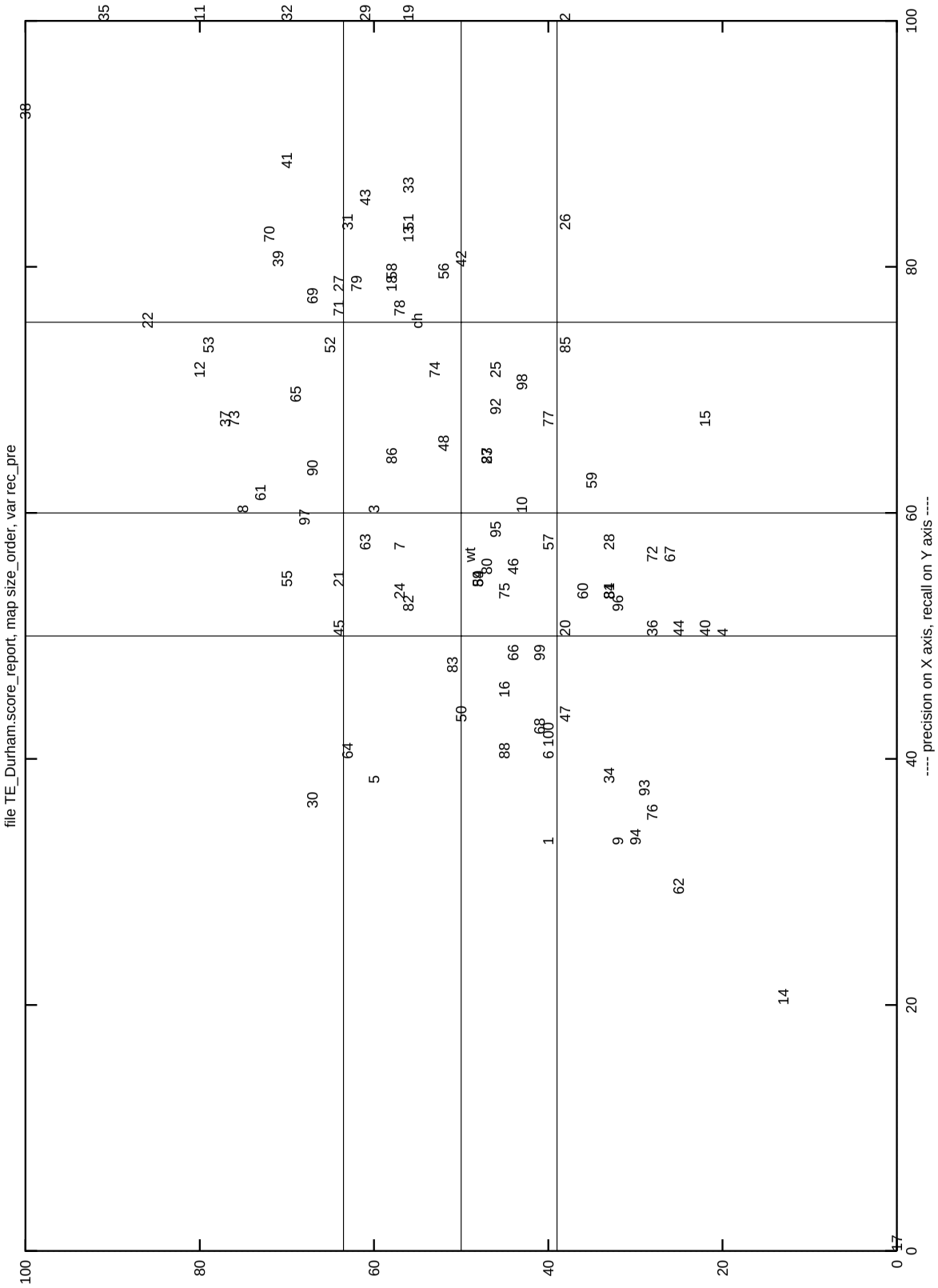


Figure 5.4: Scatter plot of recall (y) and precision (x) for Template Element

indicates that LOLITA recognises 87% of the relevant articles, but does not have equally high precision.

Recall and precision correlate highly for all 100 articles, but this does include many zeros for the irrelevant articles, so is not surprising. In the following, zeros are omitted. On only the correct relevant articles (46), the relationship is weakly negative. Against other tasks, all correlations are weak or insignificant[4]. Size had no significant effect on recall or precision. In contrast to other tasks, size did *not* significantly affect the counts (ie, not ratios) of COR, INC, or MIS. SPU correlates weakly.

5.6 Analysing the Relative Correctness

Overall score comparison and statistical significance testing can give an idea of ranking or distinguishability of systems, but as noted before, the overall scores hide a lot of detail like "how many got what". Given the different style of analysis in LOLITA, we would like evidence that LOLITA is getting features correct which few other systems get, and we expect to see relatively poor performance on the features that most get correct.

The method described in this section scores output by categorising features in a task key by the number of systems that got them correct. This contrasts with the usual 'decomposition' by template and slot[5]. We apply this method to all four tasks, then draw conclusions. A full example of the necessary calculations for correctness analysis is contained in appendix C.

Note that this study is an *exploration* of a new technique, and absolute accuracy is not guaranteed, though steps have been taken to ensure reasonable accuracy. Reasons for inaccuracy are outlined, and an estimate of its size is given.

5.6.1 Explanation of Method

Overview

The conventional scorer builds a table of counts of POS, ACT, COR &c. for each slot, and sums over all slots to produce the final totals. Instead of categorising each part of the comparison between key and response by the slot, we will categorise it by the number of systems that are correct for that slot instantiation. Just as the conventional scorer derives recall and precision from the counts it collects, then we can do the same in each correctness class; f-scores can also be calculated, using the formula in [DARPA, 1995].

So if three systems are correct for some instantiation of slot (eg ORG_NAME), then the results from that instantiation will be collected in a category for "three correct" instead of a category for ORG_NAME. Note that we may subdivide the "correctness category" by slot, eg have a subcategory "three correct on slot ORG_NAME", if we want the extra information.

Note that over-generation does not fit naturally in to this scheme. Over-generation is an error that individual systems make. Clearly, the errors of one system should not alter the counting for another system, so we reject ideas like adding a correctness point to the non-over-generating systems and incrementing the number of countable features (which has a net result of penalising the over-generator). Sometimes, two systems can produce identical over-generations. It is debatable whether both should be penalised equally, or separately. Currently, over-generation is ignored, but the relevant figure from the conventional score tables can be taken into account when interpreting the results below.

[4] As only half of the articles in the SGML test subset are relevant, only these fifteen articles are used in the statistics.

[5] We assume the template form of answers throughout, ie that CO and NE SGML is converted to templates.

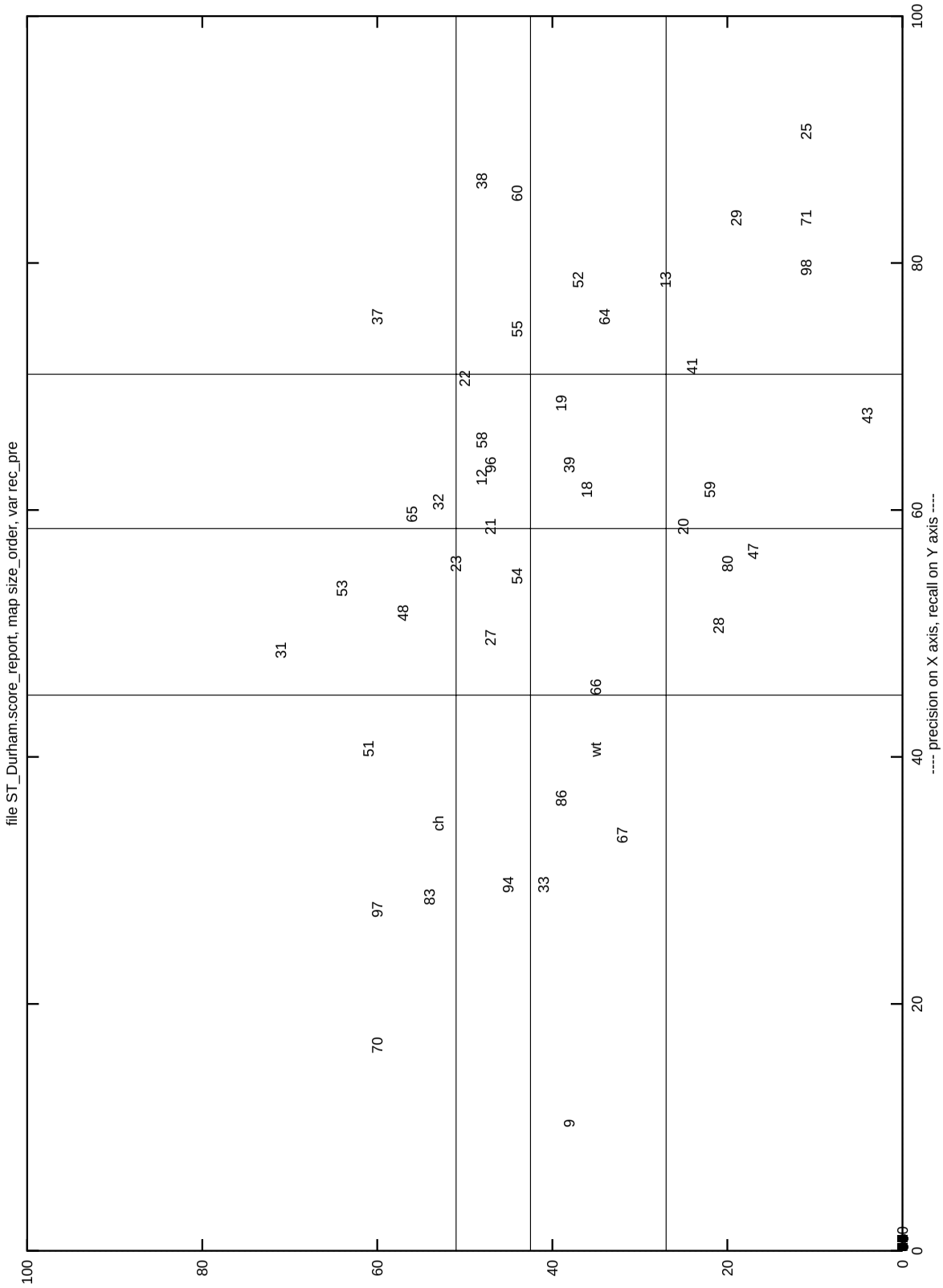


Figure 5.5: Scatter plot of recall (y) and precision (x) for Scenario Templates

Implementation

The multiple system mode of the template comparison tool has been modified to count which systems were correct for a certain feature in the task key, which were wrong, and which missed that feature. This information is indexed by the number of systems correct and by the type of slot. Over-generation is ignored, as explained above. Note that determining when two systems over-generate in a similar manner would require non-trivial extensions to the tool.

The tool is run, an article at a time, on all systems in a task. It requires a response file and scorer map for all systems. The result is several article-specific tables of raw counts. These were combined across all articles by a second `perl` script. This script calculated, for each correctness class, 'recall', 'precision', and f-score values from the totals of features correct, features attempted, and features possible. Appendix C contains a complete example of the calculations for correctness analysis.

Details of Input

The response files (and scorer maps, for template tasks) of all competitors are freely available from the MUC-6 ftp site[6]. We are not interested in the details of any other entrant's performance, so these results are treated *anonymously*. Procedurally, the input to the tool was placed in files whose name matched `<task>-<article>-<number>.<type>`. The `<number>` is an unique number for an entrant in a task, and was the only way in which systems could be differentiated in this analysis (the comparison tool maps these to a lower-case letter).

Form of Results

As mentioned in appendix C, the main result for each system is a table of counts for each correctness class. These are reduced to an array of f-score values, one for each class. With n entrants, there are $n+1$ such classes, between no-one correct and all n correct. We also have a system independent array of sizes of the correctness classes.

This information is presented graphically. In line with the observation that slot f-scores cannot be usefully compared unless scaled to indicate their *contribution* to the overall score, we multiply the f-scores in each class by the size of each class – the number of times i systems were correct. For comparison, this size is also plotted, displayed as a hollow box. The classes are shown in order, from 'no-one' correct on the left, to all correct on the right, and labelled 'C0', 'C1', ... 'Cn'.

The f-score vectors are plotted as lines to allow easy identification of systems by line styles (plotting with different point styles is not as clear because of overlaps). No meaning should be read into the transition lines. Because of the sometimes huge variety in class size, some of the graphs have logarithmic vertical axes (and an adjusted scale is shown). The TE and ST graphs contain a horizontally-lined bar which represents the average scaled f-score in a correctness class. The values do not appear to have independent interest, but they do allow us to observe when systems are markedly above or below average.

Note that all systems get a perfect score in the n-correct class by definition, and that the f-measures are undefined for the 0-correct class since because recall is undefined – no system can be correct there, by definition. LOLITA results are always the 'A' line, unless indicated otherwise.

Interpretation of Results

These graphs are a different way of looking at the MUC scores, which relies on the performance of *other* systems to provide classes of feature. We assume that the 'order' of the class (how many are correct) gives an indication of the difficulty of that class of features. We shall refer to the classes where most systems are correct as 'easy', and where few are correct as 'hard'. These terms

[6]Contact the MUC-6 organisers for details.

are loose - there is no strict delimitation of extent, so the area in between is a grey area. If a system scores better on (most) harder classes than some other system, then we interpret this as the former system performing better on harder features. Conversely, a system scoring less well on (most) easier classes means that it is performing less well on easier features.

Note that f-scores always will be high in the easier classes, as most systems will be correct in them. Similarly, f-scores will be relatively low in the harder classes if one system is performing far better. In general, there is an essential dependence between f-score and class as a consequence of our method of counting. The dependence between score and class size may require more detailed mathematical analysis, if only to eliminate the predictable part of the results. One non-mathematical possibility in the implementation is to calculate the class scores for a single system relative to other systems. That is, to ignore a system's contribution to a class. This has not been attempted yet.

No statistical significance testing is done on the results, and we cannot claim that such testing is unnecessary. We expect that it will be necessary, in the same way that significance testing is necessary for the conventional results, but we are not sure what kind of testing should be done. We leave this aspect to more expert parties. Thus, any interpretation below must be *tentative*.

When discussing the graphs, we typically read them from right to left, moving from the easier classes to the harder ones.

Accuracy

Since the method alters the way that counts are summed, the overall totals should be *identical* to those in the conventional analysis (with the exception of Coref, which uses a different counting technique). In the current implementation of the method, this is not the case; some reasons are outlined in section B.6. We have analysed the differences in totals for all tasks except Coref. For NE and TE, the differences do not exceed 3.0% in recall or precision. For ST, the difference is a slightly higher 4.3% for recall and precision. The author decided that this did not justify the extra effort required to provide more exact results. We note again that this study is experimental.

5.6.2 Named Entity Task

The two lowest-scoring systems were omitted from this analysis. The accuracy scores showed an average absolute deviation in recall of 3.0%, and in precision 1.1%. Figure 5.6 shows the log of the scaled f-measure over both the text and type 'slots'. To highlight LOLITA's performance in the presence of seventeen other systems, its graph appears as a horizontally-lined bar. There are several things to note:

- Effects of the log plot: there are 2314 possible features in the NE task[7]. The bottom half of the classes account for approximately 10% of the total, so the relative system performance patterns therein are not globally significant.
- The distribution of sizes of correctness classes: there are many features which most systems get correct, and relatively few which less than half of the systems get. Note that the size peaks at C13, corresponding to (any) five systems of the eighteen being wrong on some feature.
- The coherence of system lines: several points. Twelve systems show very similar results of high scaled f-score until C10. After C10, there is much variation in this group. Hence in the easier classes, these systems score well and score on similar things, but performance is more varied on the harder classes.

  All of the other six systems (including LOLITA) score less well on the easier classes. One (thin dotted line) is lowest in almost all classes. Two split off from the main group after C14 and get steadily worse – so until then, they are doing similar things to the main group.

[7]There are 1157 markables in the key, each with a text component and a type component, of which 44 are optional. The correctness analysis calculations includes ALL markables. See page 186 for more information.

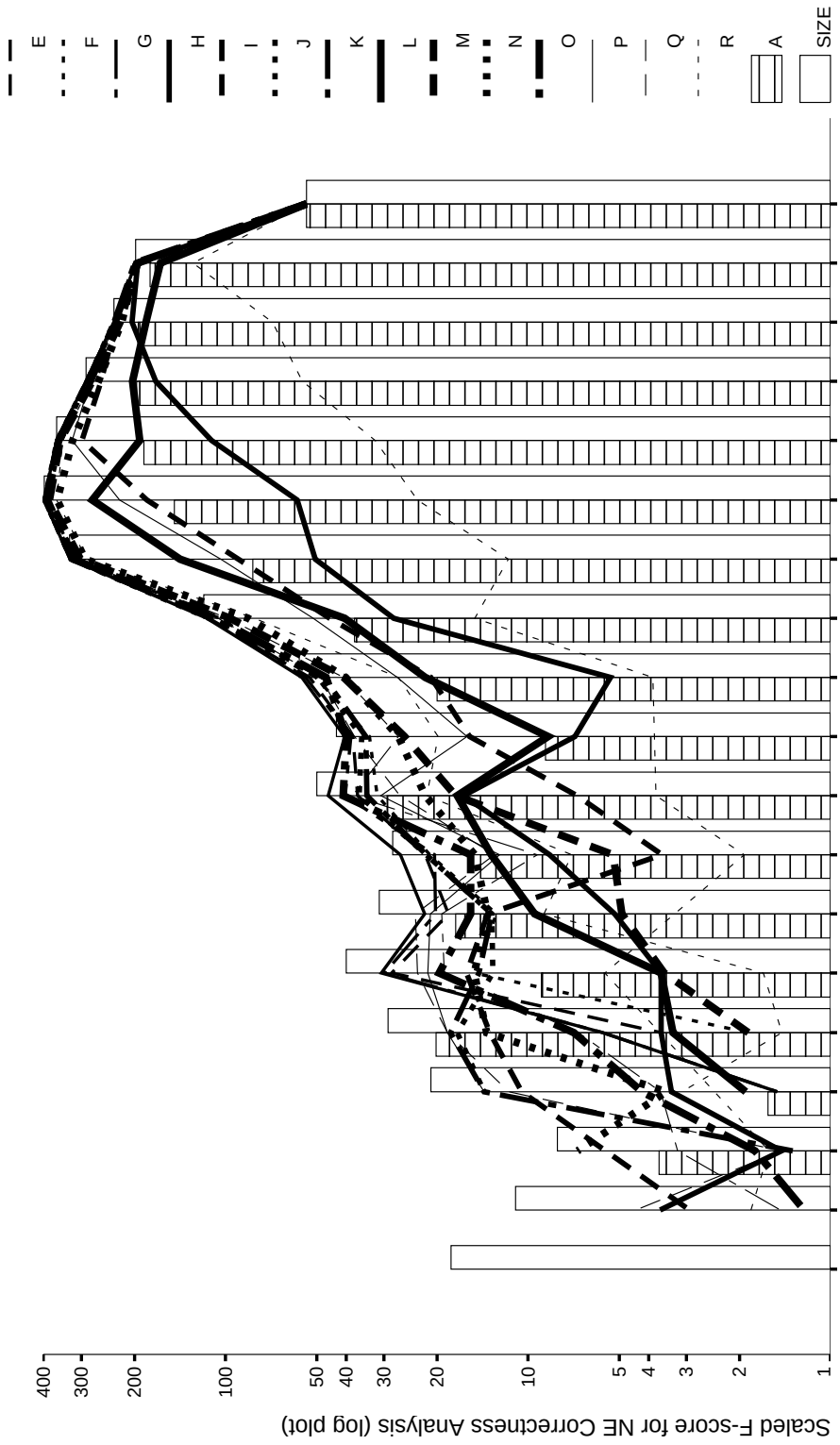


Figure 5.6: Named Entity correctness distribution: log of scaled f-measure.

Some of these six do reasonably well on the harder classes, but since these classes are relatively small, this observation lacks force. Note that most systems mirror the shape of the correctness class sizes.

- LOLITA results start to creep above the majority in the harder classes, but as noted above, the sizes of these make this observation insignificant. System 'l' follows a similar pattern.

5.6.3 Coreference Task

Coref scoring depends on *relationships* between the markups, and correctness is not a property of a particular markup. Even though markups can be converted to templates, an application of the method still has certain limitations. The f-score graphs for analysis by TEXT slot and KEY_CHAIN slot are presented together in figure 5.7 (the 'transition' line between the two groups should be ignored). All seven competitors are included. See the discussion for the accuracy checks.

TEXT slot

This indicates whether an expected markup was produced, with the proviso that our string matching is implemented as exact, and does not take MIN strings into account. Thus, correctness is given when a response's markup string is *identical* to the key's – in other words, for a subset of the results. It does not indicate whether the markup was correctly attached to other markups. We note:

- Distribution of sizes: TEXT_0 is quite big – this is because we ignore MIN string matching. However, it is not greatly larger than the other classes, so the remaining results are still worth consideration. The remaining classes are fairly evenly sized. This is interesting, when contrasted with the shape of the other task graphs. We view this as a feature of task difficulty: compared to NE, there is not a large subset of features which may be easily obtained.
- Coherence: five of the systems stay fairly close with good scores in TEXT_6 and TEXT_5. In TEXT_4, one of these (C) performs less well. Two of the five stay close all the time (B and F). LOLITA (the solid thin line) and G stay close with poorer scores.
- Overall scores: these are some way below perfect from TEXT_5 and below.
- LOLITA: results are initially poor, on the easier classes, but become relatively good on the harder classes. The score on TEXT_1 is in the middle of the field.

KEY_CHAIN

The second indicates whether the response markup was mapped into the correct key chain. This is closer to real scoring than TEXT slots, as it directly reflects an SGML markup being mapped (and thus does not suffer from our mishandling of MIN strings), but it still does not indicate a markup contributing to the score. Superficially, the features of this graph are similar to the TEXT graph. We note differences:

- Distribution of sizes: there is a smooth decrease, with a slight rise for KEY_CHAIN_0.
- Coherence: there is more in this graph. Some systems are very close over small ranges, such as C and F for most of the graph. G is interesting: it starts off similar to LOLITA, but improves to end up top in KEY_CHAIN_1. Note that C performs a lot better here.
- LOLITA: though still the lowest in most classes, the difference seems less pronounced. LOLITA is undistinguished in the harder classes.

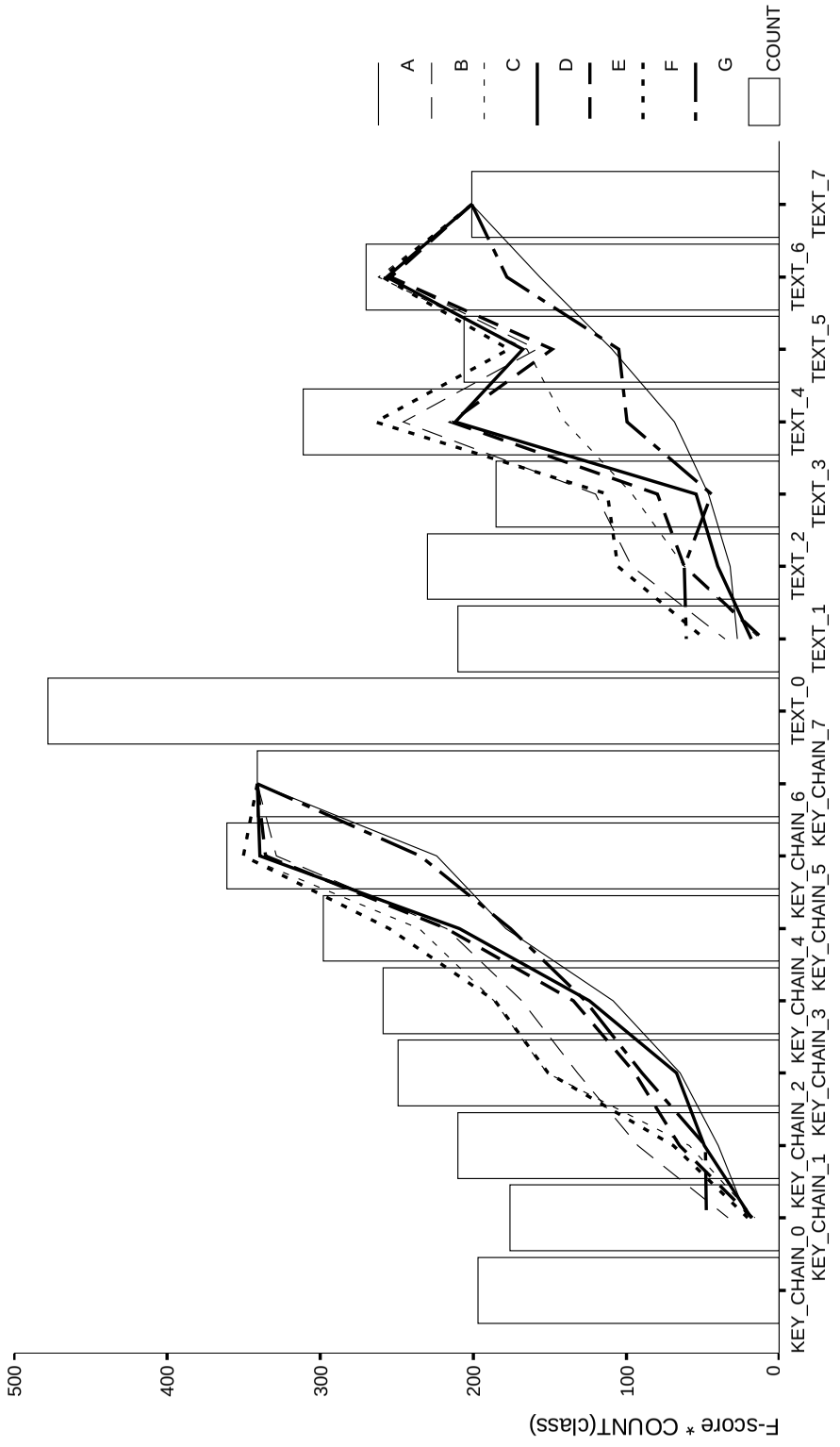


Figure 5.7: Coreference correctness distribution: scaled f-measure, for KEY_CHAIN and TEXT slots.

Discussion

It is not easy to check accuracy, as the counts we would normally use are not given in the Coref scorer output. As we noted above, coref scoring is not template-based, so our observations have little force. The limitations mean that a *subset* of coref performance is being investigated; there may be some usefulness in the observations.

TEXT slot analysis is the weaker, since MIN strings are ignored and exact string matching is used. But, this may help to illustrate the effect that MIN string matching has on the task, by comparison to the KEY_CHAIN graph. For example, system C shows better scores when counting KEY_CHAIN matches: it may prefer to output narrower strings.

Further research is needed before we can apply this analysis method to coref.

5.6.4 Template Elements

The check on correctness shows an average absolute deviation in recall of 2.3%, and in precision 2.0%. Figure 5.8 shows the log plot; a horizontally-lined bar shows the average scaled f-score per correctness class. All twelve competitors are included.

- Distribution of sizes: the graph sweeps downwards after a slight peak at C11, rising a little after C4. The pattern is similar to NE, but not as pronounced. In particular, the harder correctness classes are slightly larger. Half of the scorable features are contained in C10, C11, C12. The midpoint of the remainder occurs in C5. Observations of trends in the lower sections may lack force due to their size.
- Coherence: nine of the systems stay close until C7.
- Overall scores: After the easier classes (from C5), f-scores drop. One system, B, scores very well in the final section.
- LOLITA: starts off badly, but crosses with system L in C9 to roughly follow C in the remainder. L scores progressively worse, but the LOLITA's 'advantage' (worth around 70 points, or 2.5% f-score) is cancelled out by L's better performance in the easier classes. No other system shows a similar pattern.

5.6.5 Scenario Template

The check on accuracy shows an average absolute deviation in recall of 4.3%, and in precision 4.0%. This contains two large deviations of precision (for A and F), although most are small or zero. The problem cases are those which over-generate a lot: over-generation is currently ignored in our analysis. Taking over-generation into account for these cases reduces the disagreement. We note that the disagreement will remain relatively large because of differences in counting between the comparison tool and the ST scorer, such as the reduction in the latter of over-generation and under-generation counts by arbitrarily matching unmapped templates of the same type. The graph is in figure 5.9. All eleven systems are included. The horizontally-lined bar shows the average scaled f-score per correctness class.

- Distribution of sizes: this pattern is very interesting – the class size rises from a low C11 to a high C0. We interpret this as showing a big difference in nature of the task, compared to NE and TE.
- Coherence: there is not much outside the easy classes. The lines cross frequently.
- Overall scores: all systems show a maximum around C6-C7, and drop gradually after this. This is in contrast to the still-increasing class size, meaning f-scores are dropping relatively sharply. System F scores well overall, but it also over-generates.

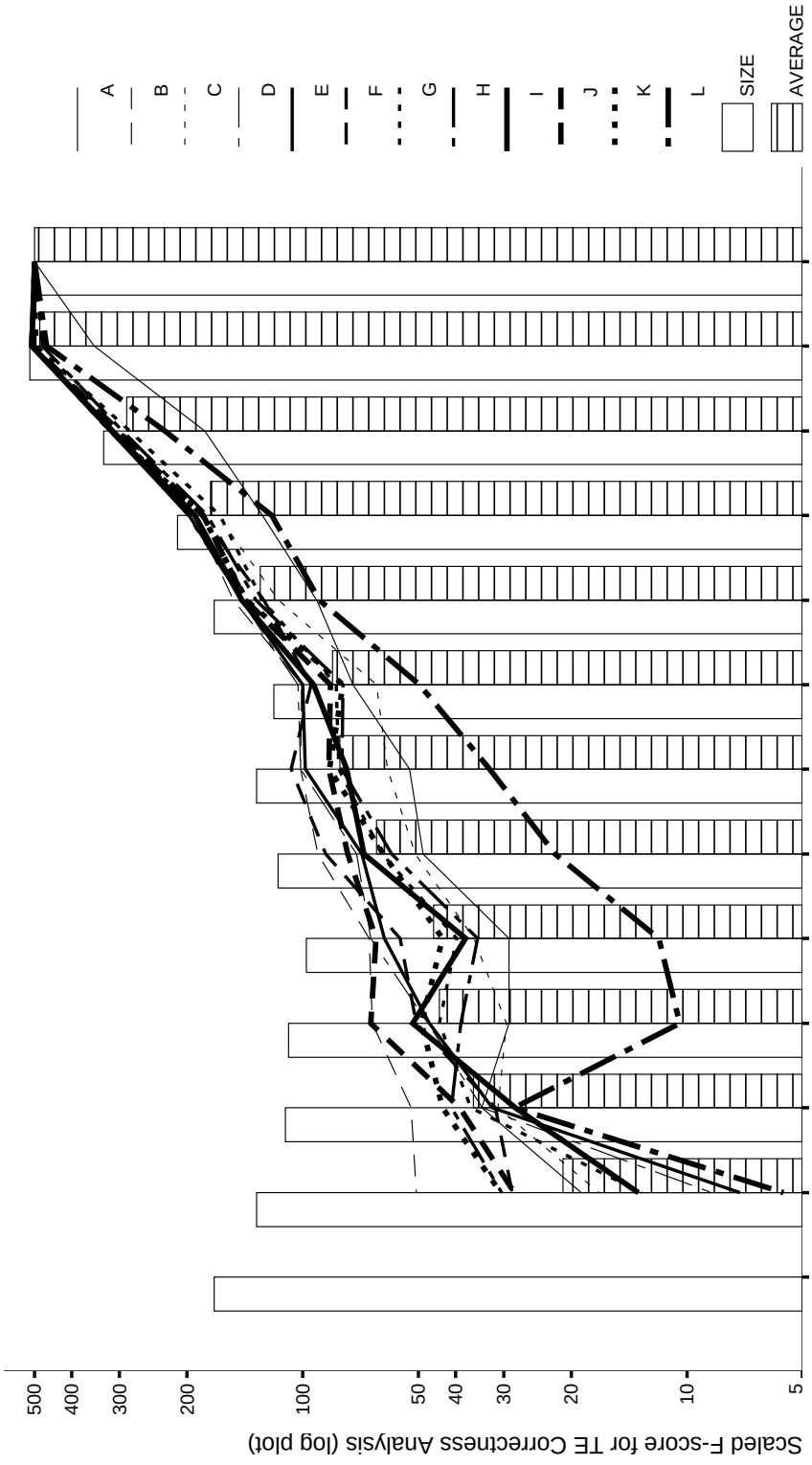


Figure 5.8: Template Elements correctness distribution: log of scaled f-measure.

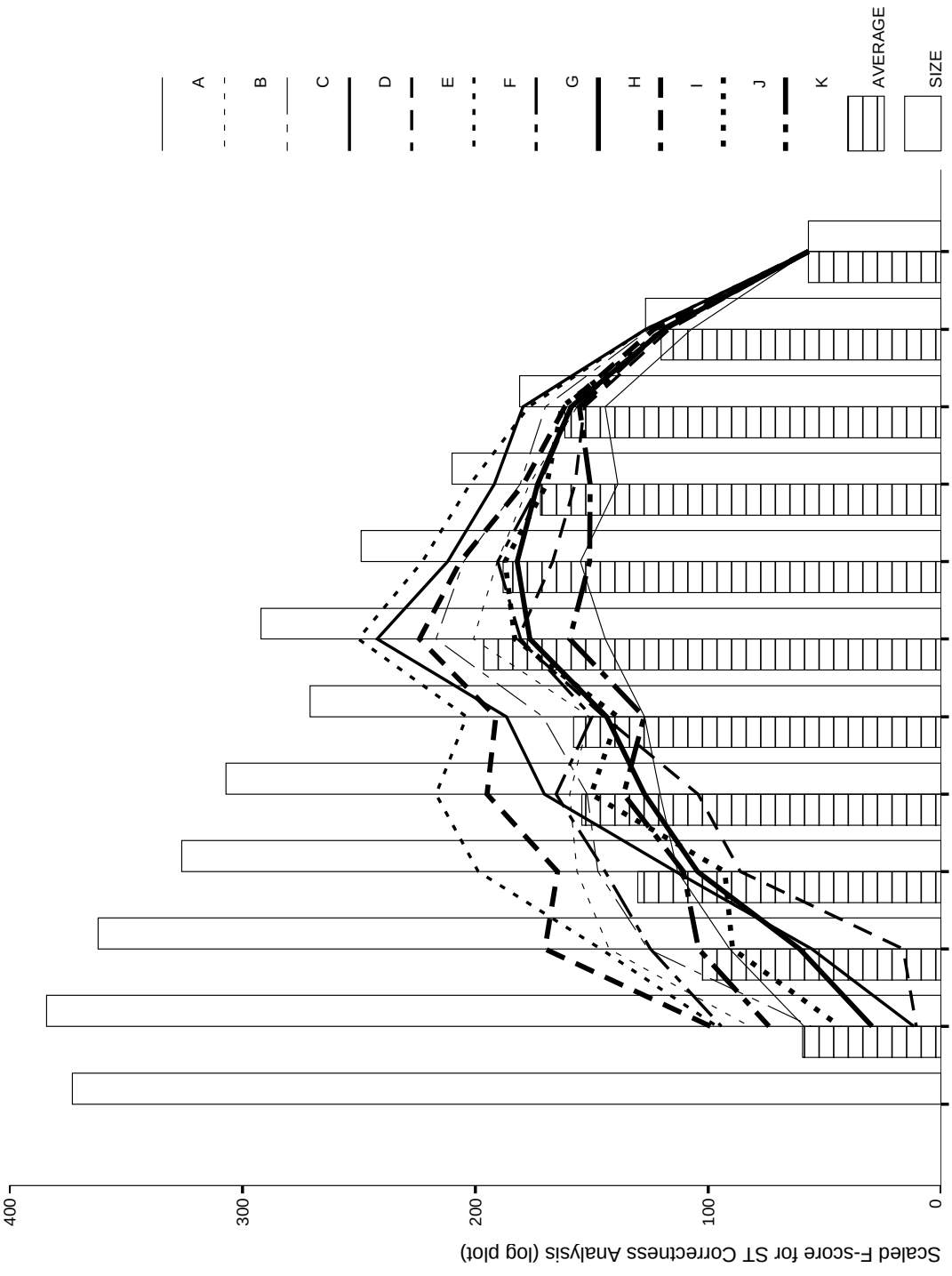


Figure 5.9: Scenario Template correctness distribution: scaled f-measure.

- LOLITA: the usual pattern is seen for easy classes, but LOLITA rejoins the main group and becomes relatively good after C5, although it never exceeds the expected average score. System 'k' shows a similar pattern.

5.6.6 Conclusions

What can be concluded from this experimental study? We first summarise what we believe the results tell us, and then consider the strengths and weaknesses of this method of analysis. The original intention was to investigate LOLITA performance against that of other systems, with a view to confirming our intuitions that that LOLITA did better on harder features of a task, but got a lot of the simple features wrong. We have certainly confirmed this, with more strength for the latter than for the former. Note that LOLITA was not always unique in having a poor 'easy' score and a relatively better 'hard' score.

We believe there are three main aspects to the graphs:

- The different distributions of the sizes of correctness classes.
- The patterns of scores within these classes.
- The coherence of systems' behaviour.

Together, we see these as indicating task difficulty and how systems are attacking the task. NE and TE show a lot of easy features which most systems get correct, and relatively few features which only a couple of systems get correct. ST shows a different pattern. There are no systems which are substantially better at harder features, though there are ones which are significantly worse on the easier features.

Generally, is this method of analysis useful? It provides a different view of the MUC scores, and shows one aspect of the relationship between systems. It sheds some light on the notion of the difficulty of a task, by examining the degree to which features are correctly produced by all competitors. The most substantial result is, we believe, the class size distribution: it is easiest to interpret, the most natural to define, and can be considered separately from the f-scores. The system-specific information is not as easy to understand, but it is still interesting. We conclude that this method is useful.

5.7 Analysing the Similarity of Systems

Identical MUC-6 scores do not imply identical output, so how systems compare by *similarity of output* is of interest. This section describes a second experimental study. The template comparison tool was further modified to record when systems performed identically on a given key feature. For each feature, the set of systems is partitioned into sets where each system is correct with respect to the key, incorrect, under-generating, partially correct &c. Within each set, each pair of systems receives a 'similarity' point for that class.

Over-generation is ignored for similar reasons to the previous section: we would have to implement code to check arbitrary systems for similarity of over-generation, which would lead towards a re-implementation of the scorer.

The end result for an article is a triangular matrix of similarity counts for each of the behaviour classes. It is triangular, as every system is compared against all others and the comparison is symmetric. Each article is processed as for the correctness analysis, and the matrices are combined over all articles.

We intend that the similarity counts should be read *relatively*, ie system X is more similar to system Y than to Z. No suitable 'absolute' suggests itself. We also can consider the inverse, of dissimilarity, by looking for low similarity counts. Triangular matrices are hard to read, so we attempt to show the results graphically. The natural style is with labelled points, but we find use of styled lines is easier to read. Each line represents one system compared against all others, as

shown on the horizontal axis. The data is sorted to make the LOLITA line increasing, hence the reordering of system names on the horizontal axis. The 'self' points have been removed to make the graphs clearer: for any system's line, these show the maximum possible in the behaviour class when comparing against that system. This graph version is not perfect; but a better method is not apparent at present.

Accuracy is still an issue. No formal checks have been done, but the counting method depends on factors similar to the correctness counting, so we assume that the accuracy figures above apply here too. Again, note that this section is an experimental study.

We present results for TE only: the current form of this method of analysis does not produce clearly significant results, and we have not been able to extend the intuitively appealing notion of similarity to a detailed interpretation scheme. We do investigate similarity within exact matches, under-generation, and incorrect fills for this task.

5.7.1 Template Elements

The table below shows the similarity counts relative to LOLITA (system 'a') across all classes of behaviour (columns), for all systems (rows). (See appendix B for explanation of symbols.) LOLITA's row shows how its performance is divided over the classes, so is the maximum possible. The other systems' rows show how many of these cases are also shown by the particular system[8]. Similar tables can be produced for each of the other systems. The TE graph is in figure 5.10. The line plotted for each system is the sums of each row in the system's table.

System	==	XX	--	++	>>	<<	.5
a	1240	272	456	66	11	19	2
b	1056	36	137	7	0	18	0
c	977	38	194	13	1	21	1
d	1085	29	178	9	0	19	1
e	1082	27	207	7	0	19	1
f	1085	65	179	12	1	18	1
g	1005	49	225	13	0	10	1
h	1012	61	184	14	1	9	2
i	1057	47	196	15	0	10	1
j	1039	37	157	8	0	13	0
k	1049	62	162	19	0	6	2
l	891	81	224	22	0	0	0

We note, for the table:

- Most other systems get around the same number of identical correct answers. In particular, there is a small amount which LOLITA gets above this number – features on which *only* LOLITA is correct.
- Correspondingly, there is a number which only LOLITA gets incorrect. It gets many more incorrect than the other systems.
- The difference on under-generation (--) is less pronounced. There is much more similarity of under-generation than of incorrect answers, for all systems.
- Note that over-generation is only counted within templates aligned with a key template. The closest system in this respect over-generates on a third of the cases LOLITA over-generates.

We note the following from the graph:

[8] There is a bug in counting for slots with alternative fills, when some of them are multiple values; normally, the value of << for the target system (here, LOLITA) should be the largest in the column. This affects only a few cases per thousand.

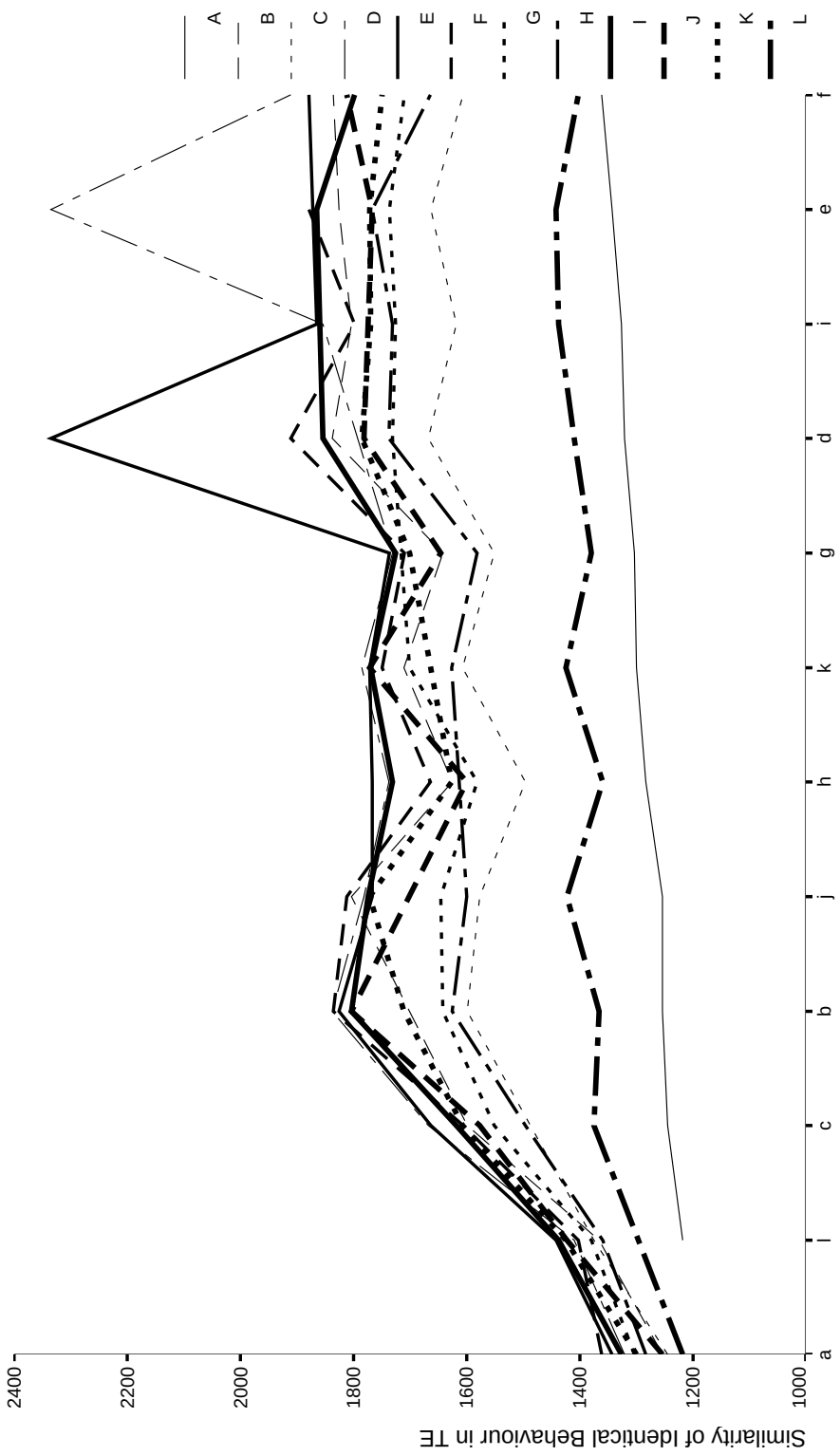


Figure 5.10: Template Elements consistency

- System A (LOLITA) is least similar to system L, and most similar to F. No system is less similar to another, than LOLITA is to any.
- D and E are very similar (they are in fact two versions of one system).
- Systems L and A are definitely less similar to the remainder than the remainder are among themselves.
- The graph is hard to interpret.

We have also analysed within-type behaviour for exact matches (figure 5.11), incorrect fills (figure 5.12), and under-generation (figure 5.13). The following is noted:

- Figure 5.11:
  - This figure is the main 'component' of figure 5.10 – its contribution is greatest. It is clearer than the overall graph.
  - Four groups can be identified: the two lower 'independents' of L and A (which is LOLITA); a middle group of C, G and H; and the remainder at the top (B, D, E, F, I, J, and K).
  - LOLITA only gets around 60% of the available similarity points in exact matches.
- Figure 5.12:
  - This figure, which shows similarity of direct mistakes, is quite convoluted.
  - The high similarity of incorrect fills for D and E is to be expected.
  - LOLITA's incorrect fills are least like D and E, but most like L.
- Figure 5.13:
  - This figure, which represents under-generation, is more ordered than the previous graph. The D and E peaks are not a surprise.
  - The LOLITA line is very high, indicating that LOLITA makes many of the under-generations that other systems do.
  - LOLITA's under-generation is most similar to systems G and L, and least similar to B.

5.7.2 Conclusions

This study was an experiment, to see what would result from an idea that had intuitive worth. The resulting matrices of similarity scores in table form are interesting. An overall picture is hard to grasp, though, as the matrices only illustrate the similarity to one target system at a time. The matrices may be most useful if comparing a specific system to all others, as we did above.

The graphical representation is harder to understand. One problem is of complexity: if we considered fewer systems at a time, and investigated the detail of types of performance (eg exactness vs under-generation), and had specific questions in mind, then we may get useful results. Splitting the detail of similarity appears fruitful: especially since similarity breaks down between systems through the variety of different errors made. Another possibility is of 'normalising' the figures in some way, to draw out the real distinctions. For example, scaling by some system-specific quantity.

In summary, the initial results are disappointing, but there may be some ways of making use of the count of similarity between systems. The method is not as interesting as correctness analysis.

Figure 5.11: Template Elements Similarity on Correct Matches

Figure 5.12: Template Elements Similarity on Incorrect Fills

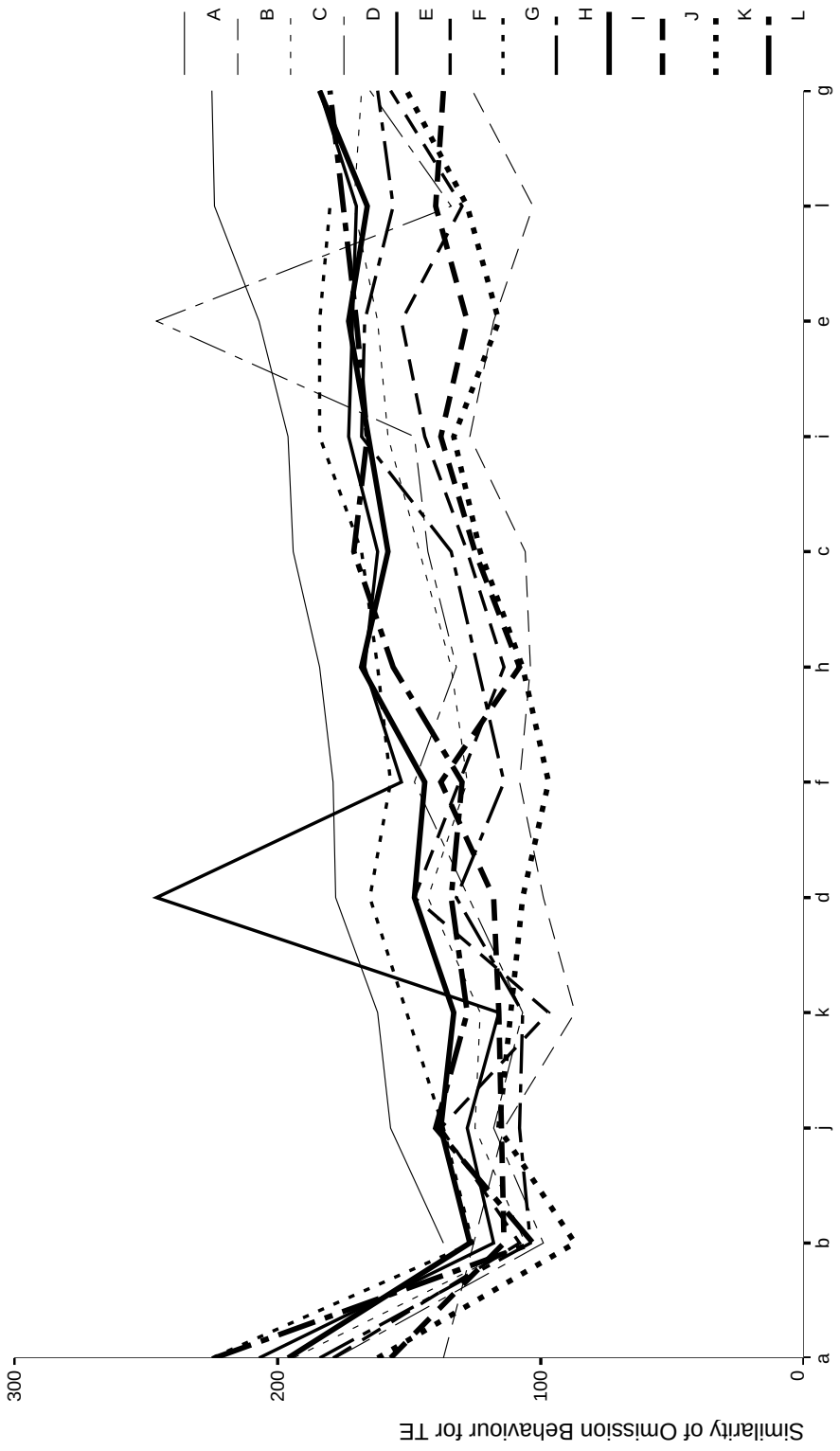


Figure 5.13: Template Elements Similarity on Under-generation

5.8 Current Performance

This section considers the change in performance one year after MUC-6. During this year, some of the serious problems in LOLITA that were highlighted by the MUC-6 work were tackled, significantly the grammar and semantic net data. This was discussed in 4.6.1, but briefly: most work was on the core, with only minor bugs corrected in the actual task implementations. The author did not contribute significantly to this post-MUC-6 work.

The conditions for the rerun are as close to the October 1995 run as possible. The same compiler and run-time parameters (eg heap size) were used. The run was made on a fast multi-processor machine. It took the equivalent of twenty hours on a single 100MHz processor. It is important to note that the changes to LOLITA were not influenced by the evaluation texts in any way, apart from the changes made to improve the walk-through article score for our MUC-6 conference paper. Therefore, this rerun is 'blind'. It is likely that this test set will remain blind, in future use.

Scores are compared to the official scores graphically, and by tests of correlation coefficient. The current scores were produced using our installations of the scorers and keys from the MUC6 ftp site. In CO and ST, we appear to be penalised for not producing optional markups and templates, hence the scores are slightly worse than they should be: this problem is being investigated at time of writing.

The graphs are of recall against precision, and display an arrow for each datum: the tail (with article size rank) is located at the previous score and the head at the new score. An improvement is an arrow pointing rightwards and upwards, with the length of arrow obviously indicating the magnitude of change. Arrows near the vertical represent a predominant change in recall, near the horizontal a predominant change in precision. The previous quartile lines are shown as left-wards and down-wards arrows, the new quartile lines as the reverse. Coincident quartile lines appear as double-headed arrows. Identification of the quartile (eg, first or second) must be done by context (ie, relative to the other quartile lines). The author wrote a `gnuplot`-based utility to draw these graphs, adapted from the utility of section 5.1.1.

Some of these graphs are cluttered, but note that only *some* of the lines are significant. We expect general improvement in the scores, so are only interested in large improvements and definite deteriorations. These are easy to pick out. One way of un-cluttering the graph is to scale each article by a statistic derived from it. For example, scaling by the number of POSsible correct features highlights the bigger changes and gives an idea of an article's contribution to the global score. One or both variables in the plot could be scaled, although changes of gradient after scaling by unequal factors could mislead. Scaling one variable only can reveal more detail in the un-scaled variable for smaller articles (else, the results for the high-scoring smaller articles and the lower-scoring big articles appear in the same area). In these graphs, features such as large articles scoring very badly are easily visible as arrows with high size ranks beneath the arrows for smaller articles. An example of only recall scaled by POS is given in TE below.

5.8.1 Named Entity

SLOT	POS	ACT	COR	PAR	INC	MIS	SPU	NON	REC	PRE	UND	OVG	SUB	ERR
enamex	942	907	786	0	0	156	121	0	83	87	17	13	0	26
type	942	907	668	0	118	156	121	0	71	74	17	13	15	37
text	942	907	736	0	50	156	121	0	78	81	17	13	6	31
timex	111	97	86	0	0	25	11	0	77	89	23	11	0	30
type	111	97	86	0	0	25	11	0	77	89	23	11	0	30
text	111	97	81	0	5	25	11	0	73	84	23	11	6	34
numex	93	85	84	0	0	9	1	0	90	99	10	1	0	11
type	93	85	84	0	0	9	1	0	90	99	10	1	0	11
text	93	85	82	0	2	9	1	0	88	96	10	1	2	13
ALL OBJECTS	2292	2178	1737	0	175	380	266	0	76	80	17	12	9	32

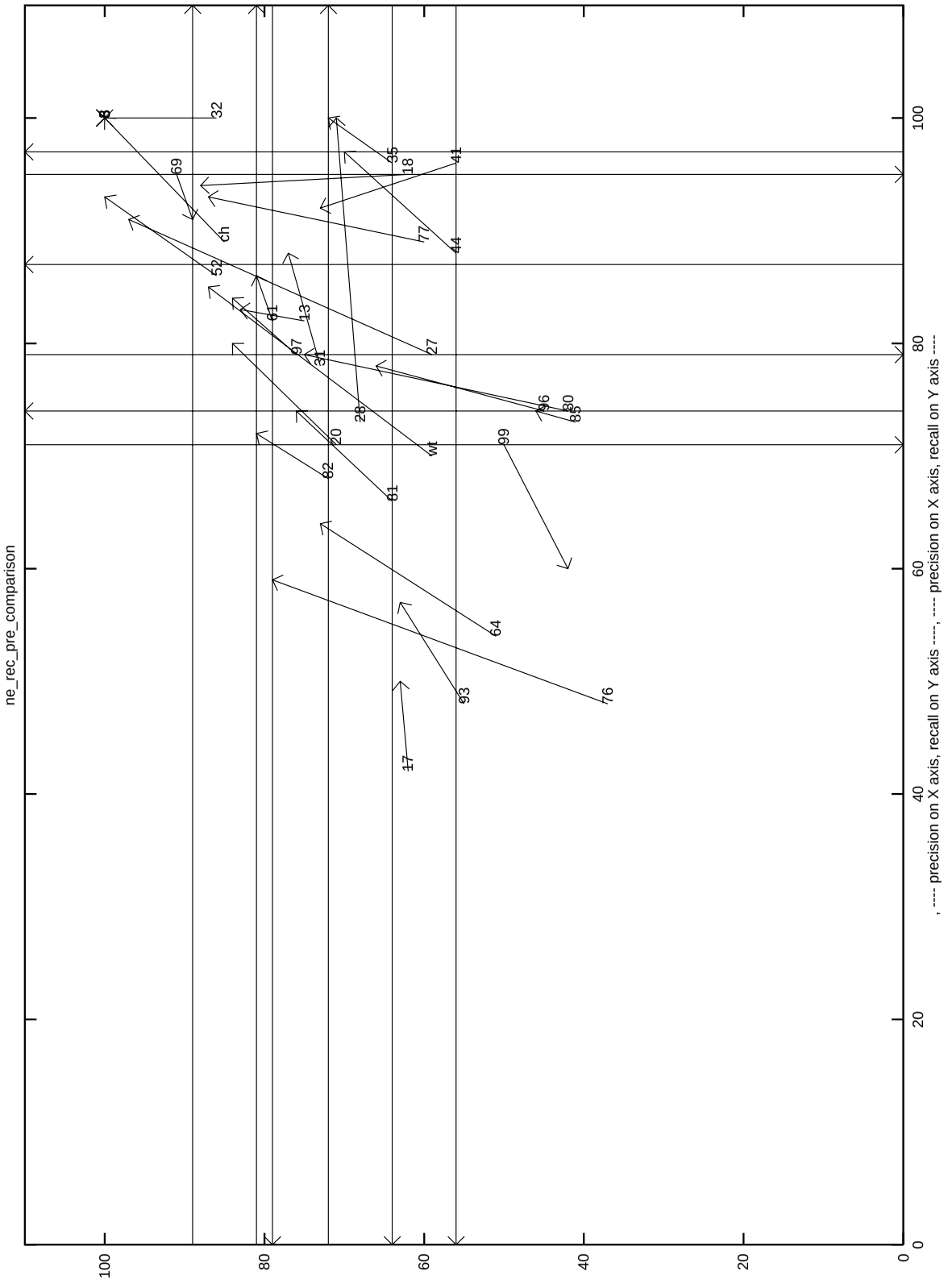


Figure 5.14: Change in recall (y) and precision (x) scores in Named Entity

```
-----------------------+------------+-------------+------------------------
                                                 P&R      2P&R      P&2R
F-MEASURES                                     77.72     78.93     76.55


 * * * TASK SUBCATEGORIZATION SCORES * * *
-----------------------+------------+-------------+------------------------
SLOT           POS   ACT| COR PAR INC | MIS  SPU  NON| REC PRE UND OVG SUB ERR
-----------------------+------------+-------------+------------------------
Enamex:
  organizati   459   459| 291   0   88|  80   80    0|  63  63  17  17  23  46
  person       372   343| 305   0   13|  54   25    0|  82  89  15   7   4  23
  location     111   105|  72   0   17|  22   16    0|  65  69  20  15  19  43
-----------------------+------------+-------------+------------------------
Timex:
  date         111    97|  86   0    0|  25   11    0|  77  89  23  11   0  30
-----------------------+------------+-------------+------------------------
Numex:
  money         76    68|  67   0    0|   9    1    0|  88  99  12   1   0  13
  person        17    17|  17   0    0|   0    0    0| 100 100   0   0   0   0
-----------------------+------------+-------------+------------------------

 * * * DOCUMENT SECTION SCORES * * *
-----------------------+------------+-------------+------------------------
SLOT           POS   ACT| COR PAR INC | MIS  SPU  NON| REC PRE UND OVG SUB ERR
-----------------------+------------+-------------+------------------------
HL             136   120|  91   0   11|  34   18    0|  67  76  25  15  11  41
DD              60    60|  60   0    0|   0    0    0| 100 100   0   0   0   0
DATELINE        52    52|  52   0    0|   0    0    0| 100 100   0   0   0   0
TXT           2044  1946|1534   0  164| 346  248    0|  75  79  17  13  10  33
-----------------------+------------+-------------+------------------------
```

The f-score has increased by 10 points, with the recall improvement being the larger component. The biggest (and most important) change has been in ENAMEX, with similar improvements in both recall (15 points) and precision (6 points) of text and type. The recall improvement is shared across the three types. For precision, organisation and location improves more than person – but the latter is already quite good. NUMEX recall has improved, with a slight drop in precision – with greatest improvement in the percent type (this is marked as 'person' in the table owing to a scorer bug). TIMEX text has improved in both recall and precision, whereas type recall increases by one point and precision by five points. By section, headline recall has improved by 20 points with a slight drop in precision. Datelines, already scoring well, are now fully correct, and DDs stay at full score. The main text picks up 14 recall and 6 precision points.

Figure 5.14 shows the score changes. All articles except four show an increase in both recall and precision. The improvements in recall (with a lesser precision increase) of the large articles ranked 76, 80, and 96 are encouraging. The smaller articles 18 and 41 lose a few points of precision whilst gaining more in recall. Articles 69 and 99 lose both, dropping more in precision than recall. The recall inter-quartile region shifts upwards by a good amount. For the precision range, the median and lower quartile increase but the upper quartile drops slightly because three of the decreases were from the top quartile and were not balanced by analogous improvements. Unsurprisingly, correlation against the previous scores is high for both recall and precision. The new scores show a fair correlation of recall and precision. We conclude that NE improvement is shared over most articles.

5.8.2 Coreference

```
Coreference Totals:     Recall: 622/1627 =   0.38
                     Precision: 622/1133 =   0.55
```

Recall may be a point or two higher than shown due to the problem with optionals mentioned before. Precision has improved by 11 points as correctness increases and over-generation decreases. Graphically (figure 5.15), only half of the articles show no decrease in recall or precision.

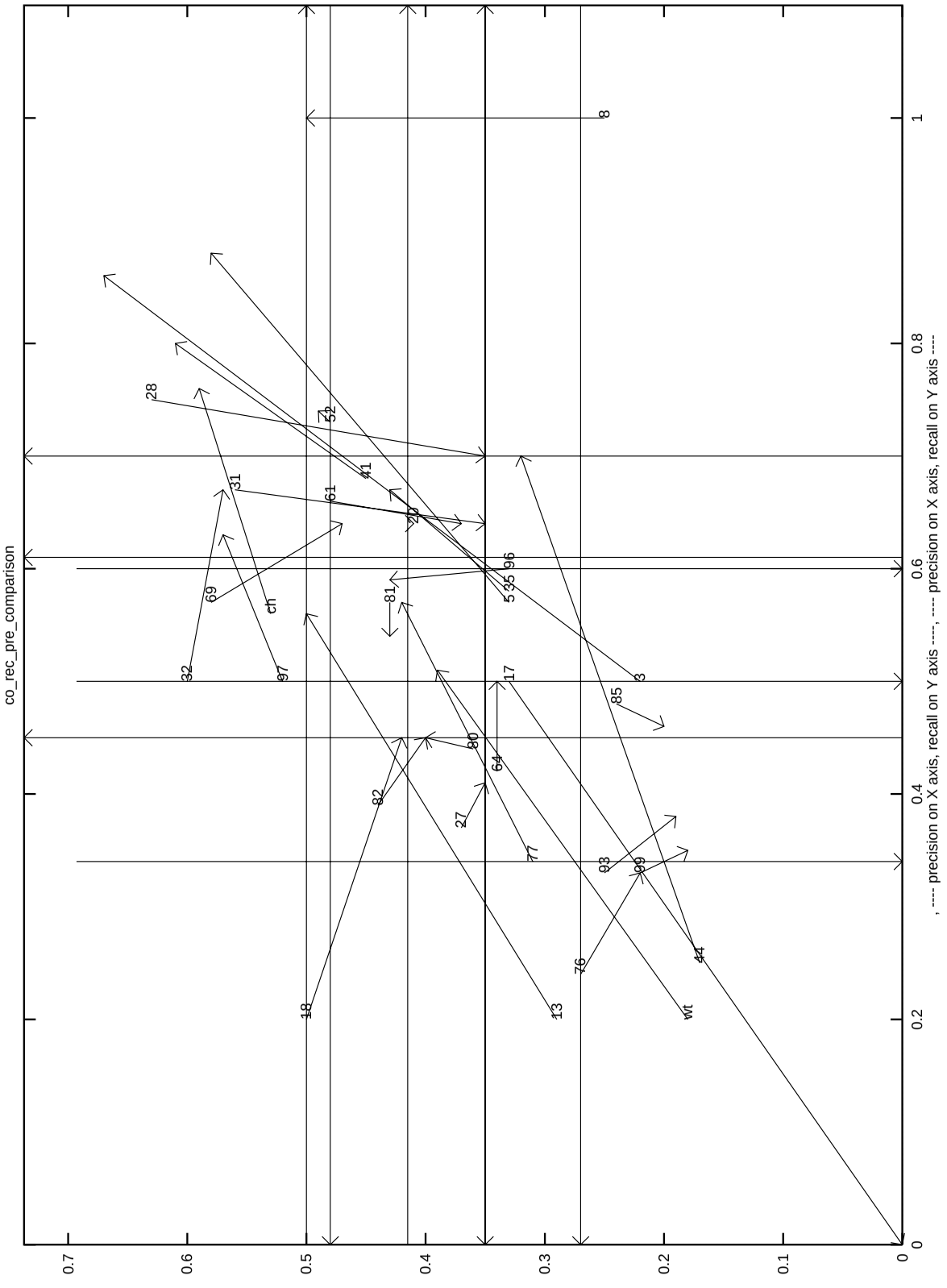


Figure 5.15: Change in recall (y) and precision (x) scores in Coreference

There are: five decreases in both variables (17, 28, 31, 61, 85); one decrease in precision only (81); seven losses in recall which gain in precision (18, 27, 32, 69, 76, 82, 93) - five of which show similar gradients of gaining more than losing – the exceptions being 69 and 93; and one loss of precision to gain recall (96). The biggest gains are usually in small articles.

Precision correlates fairly well, but recall shows only a very weak relationship with the old scores. The two variables show a moderate relationship in the new scores. In conclusion, the scores vary a lot, without consistent improvement as in NE: gains in some articles are balanced by losses in others, hence the small change in scores. But, LOLITA has generally improved in precision.

5.8.3 Template Elements

SLOT	POS	ACT	COR	PAR	INC	MIS	SPU	NON	REC	PRE	UND	OVG	ERR	SUB
organization	606	572	458	0	48	100	66	0	76	80	17	12	32	9
name	547	565	293	0	177	77	95	1	54	52	14	17	54	38
alias	170	120	78	0	3	89	39	336	46	65	52	33	63	4
descriptor	236	102	28	0	32	176	42	267	12	27	75	41	90	53
type	606	572	440	0	66	100	66	0	73	77	17	12	35	13
locale	115	24	8	0	13	94	3	232	7	33	82	13	93	62
country	116	24	21	0	0	95	3	231	18	88	82	13	82	0
person	496	460	354	0	68	74	38	0	71	77	15	8	34	16
name	496	460	333	0	89	74	38	0	67	72	15	8	38	21
alias	170	159	127	0	2	41	30	214	75	80	24	19	37	2
title	166	172	153	0	0	13	19	216	92	89	8	11	17	0
ALL OBJECTS	2622	2198	1481	0	382	759	335	1497	56	67	29	15	50	21

F-MEASURES	P&R	2P&R	P&2R
	61.45	64.88	58.37

The f-score improvement is almost 8 points, with recall and precision increasing equally. Both organisation names and aliases gain 10 points of recall and around 14 of precision. Attempts at descriptors increases four-fold, but with less precision. The organisation type score is similar to before. Country scores also improve, whereas locales drop precision with a small increase in recall. Person performance has increased slightly more: name recall is up 15 points (gaining 10 points in precision), and the other slots 11 points. Alias precision increases slightly and title precision is unchanged. This is not surprising as title generation has fairly clear cues and strict rules, so mistakes are not likely.

Figure 5.16, with one hundred arrows, is not easy to read, but some interesting features may be seen. Around 25 articles have arrows indicating a trade of recall and precision, and around 6 are losses of both (these are rough counts from the graph). Concentrating on the 50 larger articles, there are five changes in recall or precision only, 11 tradeoffs, and 4 decreases in both variables. There is no noticeable bias in the distribution of these. Figure 5.17 shows more detail. Each recall value has been multiplied by the POS count for each article, leaving the precision values as before. Note that article 33 does well, that six smaller articles get perfect precision, and articles like 100 and 96 score badly given their size. Article 89 is a significant loss because of its size. The quartile lines have all shifted up by similar amounts. Old scores correlate moderately with the new scores, precision showing a slightly stronger relationship. A similar strength of relationship is seen between recall and precision in the new scores.

5.8.4 Scenario Templates

SLOT	POS	ACT	COR	PAR	INC	MIS	SPU	NON	REC	PRE	UND	OVG	ERR	SUB
template	53	68	43	0	0	10	25	22	81	63	19	37	45	0

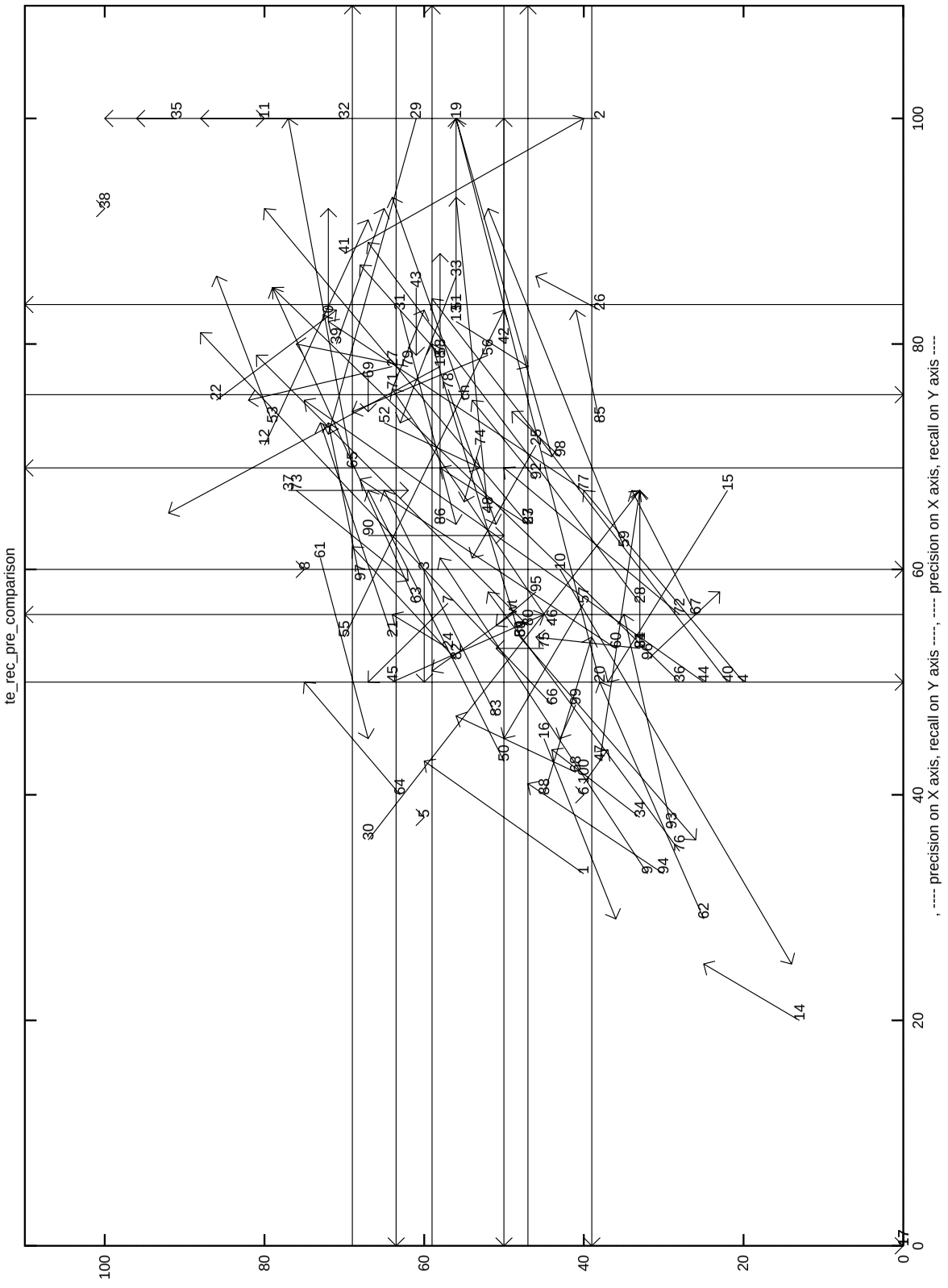


Figure 5.16: Change in recall (y) and precision (x) scores in Template Elements.

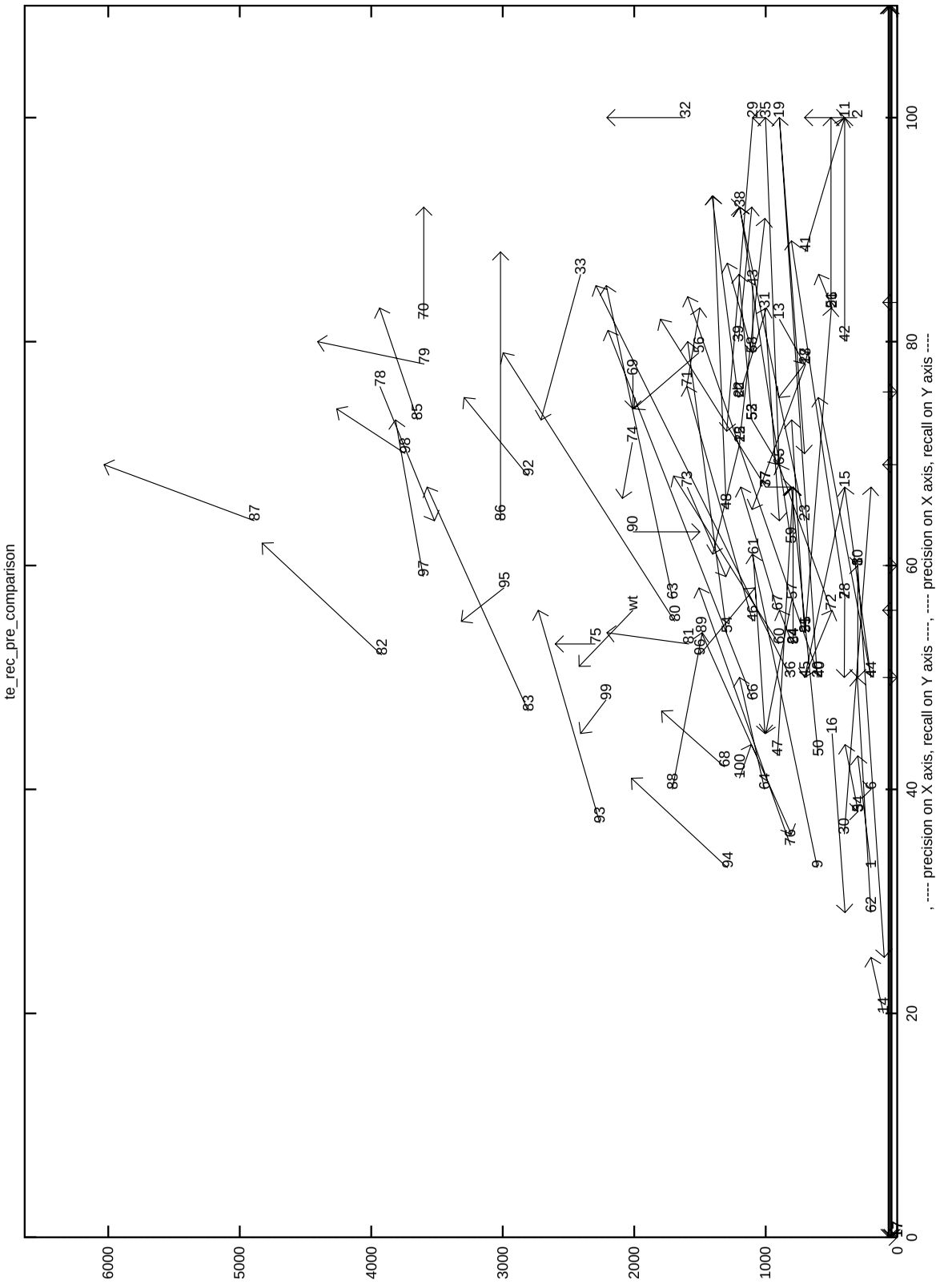


Figure 5.17: Recall scaled by POS (y) and precision (x) scores in Template Elements.

content	182	123	90	0	7	85	26	0	49	73	47	21	57	7
succession_e	192	173	100	0	7	85	66	0	52	58	44	38	61	7
success_or	192	104	44	0	23	125	37	0	23	42	65	36	81	34
post	192	173	33	0	74	85	66	0	17	19	44	38	87	69
in_and_out	258	399	110	0	35	113	254	0	43	28	44	64	79	24
vac_reason	192	173	53	0	54	85	66	0	28	31	44	38	79	50
in_and_out	260	399	176	0	8	76	215	0	68	44	29	54	63	4
io_person	260	397	154	0	30	76	213	0	59	39	29	54	67	16
new_status	260	399	115	0	69	76	215	0	44	29	29	54	76	38
on_the_job	260	399	85	0	99	76	215	0	33	21	29	54	82	54
other_org	177	0	0	0	0	177	0	51	0	0	100	0	100	0
rel_oth_or	177	0	0	0	0	177	0	51	0	0	100	0	100	0
organization	113	70	40	0	0	73	30	0	35	57	65	43	72	0
name	110	70	25	0	14	71	31	0	23	36	65	44	82	36
alias	65	29	18	0	1	46	10	8	28	62	71	34	76	5
descriptor	65	15	6	0	2	57	7	12	9	40	88	47	92	25
type	113	70	40	0	0	73	30	0	35	57	65	43	72	0
locale	42	1	0	0	0	42	1	9	0	0	100	100	100	0
country	42	1	0	0	0	42	1	9	0	0	100	100	100	0
person	134	211	97	0	13	24	101	0	72	46	18	48	59	12
name	134	211	90	0	20	24	101	0	67	43	18	48	62	18
alias	86	112	65	0	0	21	47	27	76	58	24	42	51	0
title	82	111	70	0	0	12	41	27	85	63	15	37	43	0
ALL OBJECTS	2889	2787	998	0	428	1463	1361	194	35	36	51	49	77	30
TEXT FILTER	53	68	43	0	0	10	25	22	81	63	19	37	45	0

F-MEASURES	P&R	2P&R	P&2R
	35.17	35.55	34.79

Overall, recall and precision (and hence the f-score) have all increased by around two points, which is quite disappointing. Examining the difference in score tables, it appears that recall is traded for precision on many slots. There are no large changes in any feature, so the balance of small gains and small losses leads to the small change overall. Externally most significant is the 10% reduction in (correct) succession events. Person-related features (eg IO_PERSON, the PERSON template) have improved, maybe a consequence of the better name handling noted in the previous tasks.

The graph (figure 5.18) confirms the tradeoff: eight articles improve (2, 12, 33, 40, 59, 83, 86, 91), ten get worse (20, 28, 54, 55, 58, 60, 65, 70, 96, 97), and the remaining 31 trade recall and precision in varying degrees. The arrows to the origin are losses of score; from the origin, scores are gained on previously non-scoring articles. This is very significant for the large articles (55, 58, 70), although the loss of recall could be balanced by an increase in precision, as the prior over-generation cannot now occur. For precision, all three quartile lines drop, indicating a general decrease in *individual* precision. Note that overall precision is weighted by article content, so a large article's improvement will cancel out several smaller articles' decrease. For recall, the upper quartile line is constant, and the other two rise a few points.

In correlation, precision shows a very weak relationship to the old precision scores, with recall showing a weak relationship. No significant relationship exists between the new recall and precision. For these tests, the articles scoring zero in both evaluations were discarded.

5.8.5 Conclusions on Current Performance

Good increases were seen in NE and TE, with a fair increase in precision for Coref, but disappointingly little improvement in ST. Informally, basics such as name handling have improved and the system is more conservative – so recall is traded for better precision. These scores still leave us *below* the main group in all tasks. Of course, our competitors may have improved at the same time.

It is the author's opinion that the post-MUC work has been driven too much by scores and not by examining the detail of internal performance. In particular, losing some functionality in one area and gaining a bit more in another has been interpreted as an improvement. As noted

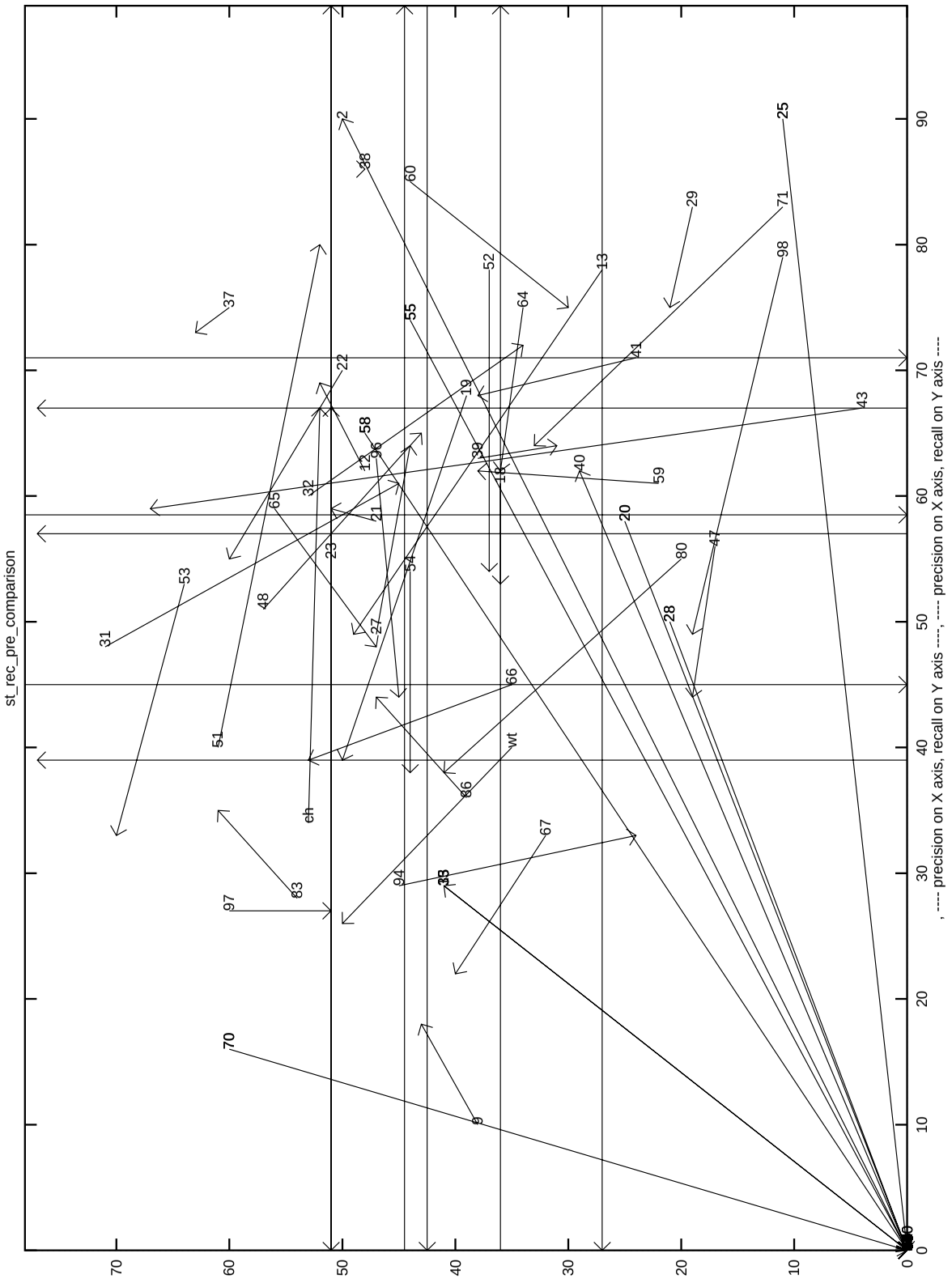


Figure 5.18: Change in recall (y) and precision (x) scores in Scenario Templates.

before, some of the loss could be explained by no longer producing poor output on which the scorer is generous – but most of it is a genuine loss. Hence, closer attention to the fine detail should result in even better scores.

5.9 LOLITA Internal Performance

So far, we have not considered the success of the LOLITA's core analysis. As noted in section 1.4.2, no quantitative or formal methods are used in development work on the system to determine the 'goodness' of core analysis. No serious effort has been made in adapting the metrics of others, or in developing our own. One unfortunate consequence of the informality is that we have no methodology for detecting a "good analysis".

We could perform a manual, qualitative analysis of the internal results and compare elements found to determine which looks interesting, but this is very laborious and not expected to be worth the effort, given the overall weaknesses observed in the previous chapter. For these reasons, we make no further comment.

5.10 Conclusions

As noted in the introduction, we cannot draw great significance from an analysis of the score tables. LOLITA clearly performs less than perfectly, as do all systems in MUC-6. Against these systems, if we assume that f-scores can be compared directly[9], LOLITA is last in all tasks except NE, where it is fourth from bottom.

The current scores show some improvement in NE, TE, and Coref, but little in ST. Examination of the finer detail shows LOLITA losing marks in one area but picking up new marks in another. If a quantitative assessment of score changes is used (eg, based on the template comparison tool), detection of these losses could result in better scores.

Two novel methods of analysing MUC-6 output were presented, which made use of the comparison tool and the output of our competitors. The first categorises scorable features by how many systems got them correct, and the second considers the similarity of behaviour between pairs of systems. The results of the latter were found hard to interpret; more work is needed. The former is much more promising: it provides quantitative information about the nature of a task, and indicates interesting trends in performance. In particular, it confirms that LOLITA does worse on the features that most systems get correct, but relatively better on the harder features.

Pragmatically, we can ask, what can we *do* with the scores? There is no simple link between score category and system component, so they do not suggest a program of work to bring LOLITA up to the level of its competitors. The results of relative correctness analysis indicate that we are losing a lot of easy marks, and informal analysis suggests this is due to simple holes in our rules or correctable bugs. The template comparison tool can be used to detect these at the article level, using the output files from the other competitors, although this would mean developing LOLITA on a test set which was intended for blind use.

Finally, what does MUC-6 tell us about LOLITA? The basic scores are not impressive, but then again, LOLITA was not designed to perform those tasks alone. Bad results are not immediately damning, though they do indicate definite weaknesses in a purported general system. Participation in MUC-6 has helped to identify some serious weaknesses, and helped to provide us with the infrastructure to reduce them. But in the end, MUC-6 tests only a subset of LOLITA: more thorough evaluation is required.

[9] Technically, we should consider the statistical significance of the difference in performance, as in [Chinchor, 1995b], but the results therein support our informal statements.

Chapter 6

Evaluating MUC-6

The main part of this chapter is a general discussion of the goals and design of MUC-6, its methodology and infrastructure, and then of the tasks in the final evaluation. We end with some suggestions for future MUC events, then conclusions. Some familiarity with the MUCs is assumed in this chapter.

There are good papers in the proceedings ([Sundheim, 1995] and [Grishman and Sundheim, 1995]), which are summarised in the first section. The summaries also serve to introduce some important concepts and terminology before the main discussion. This chapter adds detail to these papers, but also provides an independent developer's examination of the event, particularly in the methodology and infrastructure. Material from the papers is explicitly marked.

6.1 The Official View of MUC-6

6.1.1 The Design and Goals

[Grishman and Sundheim, 1995] (henceforth GSD) discusses the planning behind MUC-6. This immediate section is a precis of that material.

The MUC conferences are "designed to promote and evaluate research in information extraction", and the main innovation in MUC-6 was in the range of possible tasks (competitors need not enter all tasks). Several goals were identified, of which the major ones were:

- demonstrating domain-independent component technologies of information extraction which would be immediately useful. This was the idea behind the Named Entity task.
- encouraging work to make information extraction systems more portable (between domains of application). "The committee felt that it was important to demonstrate that useful extraction systems could be created in a few weeks". This was done by splitting the template task into "Template Elements" (to identify organisations and people, with attributes thereof), and a "Scenario Template" task (to identify *relevant* people and organisations, and the relationships between them, given a scenario only a few weeks before the evaluation deadline).
- encouraging work on "deeper understanding" (their quotes), to act against the tendency of systems to rely on local pattern matching techniques. This originally had the form of three tasks, collectively called 'SemEval', which were "intended to measure aspects of the internal processing of an information extraction or language understanding system":
  - **Coreference**: marking coreferential relations between noun phrases of kinds including identity, set-subset, and part-whole. Due to problems in formulating reliable annotation guidelines, only identity relationships were retained in the final task definition.

Even then, "it proved remarkably difficult to formulate guidelines which were reasonably concise and consistent".

- **Word Sense Disambiguation**: to determine the WordNet [Miller, 1990] sense of all open class words. Consistently tagging words in a trial annotation was found very difficult, given the fine distinctions made by WordNet.
- **Predicate-argument Structure**: to create a tree which represented grammatical-functional relationships between constituents. As before, reaching agreement on annotation guidelines proved difficult for constructs above the simplest.

The core version of coreference was seen as the most workable task and the most universal of this 'ambitious' programme, so the remainder was dropped. It was admitted that even this task makes presumptions about the internals of a system. However, the presumption was currently not too controversial, participation was to be optional, and it was argued that including such a novel task was worthwhile as it would yield new information for use in future events.

Like MUC-5, which used three sources of news article, multiple sources were considered, but a single source was decided upon: "multiple sources, with different formats and text markup, would be yet another complication for the participants at a time when they were already dealing with multiple tasks".

GSD claims that "many, though not all" of the original goals were met, and that results were comparable to prior MUCs in the Scenario task (none of the other tasks had suitable precursors), despite the shorter development time allowed.

6.1.2 Overview of the Results

This section summarises the general points of [Sundheim, 1995] (henceforth SO). Several comparisons with MUC-5, in particular some comments from [Sundheim, 1993], have been added by the (thesis) author. The detailed comments on specific tasks are discussed in section 6.3

One notable point is that the analysis of results in SO is quantitative and is mainly in terms of the f-measure (a weighted combination of recall and precision). This contrasts to MUC-5, in which the error-based statistic ERR was the official metric (ERR was the average error per slot fill, ie the count of errors divided by the number of distinct slots in the key and response). The change is probably due to the observation of little significant difference in the statistical analyses of ERR and f-measures in MUC-5. Occasionally, sub-parts of a task (such as performance on specific slots) are considered, but the main consideration is of the overall performance. There is no consideration of scores on individual articles (eg scatter plots).

Analysis in MUC-5 also considered the different classes of error (UND, OVG, SUB), which is missing in MUC-6. Reasons for this could include the greater number of more diverse tasks, which limits comparability. The MUC-5 analysis was interesting, particularly from the point of view of task definition: the classes of errors were shown to be related to the kinds of slots (eg fixed fill, variable fill). The author has performed similar analyses on LOLITA's MUC-6 results, but has found nothing worth adding to the discussion.

Of the design: the range and nature of the tasks "implicitly encouraged sites to design general-purpose architectures that allow the production of a variety of types of output from a single internal representation". This would promote use of many analysis techniques in all tasks: "Even the simplest of the tasks, Named Entity, occasionally requires in-depth processing", citing the non-markable weight "60 pounds" which six systems (a third) marked as monetary. The author notes cases of task-specific ambiguity like this are rare, and not indicative of the requirements of tasks as a whole. SO mentions several such examples as an indication of task difficulty, but does not indicate the accuracy of systems on such features. It is possible that no system could be getting them correct[1].

[1] The template comparison tool (Appendix B) allows easy checking of such assertions.

Of the corpus:

- Style: a single source of journalistic articles was used, with a bias towards financial subjects in addition to the requirement of half being relevant to the Scenario task. Additional biases included the high number of people and organisations due to the management *succession* topic of the Scenario task.
- Size: only 30 articles were used in the SGML tasks, plus an extra 70 in the template tasks.
- Case: use of case was very consistent, allowing reliable detection of proper names in the main text. One system reports a drop of around 10 points in f-measure with the test set converted to upper case. The author notes that other reliable cues such as non-standard words and suffices of a possible name (eg 'Corp'), in addition to sharing this information across occurrences of part of the hypothesised name, could be preventing further loss. It would be an interesting experiment to re-test after eliminating suffices and replacing strange words with common surnames. This idea is examined further in section 6.4.4. Note that the task guidelines sometimes make use of case distinctions, and that languages vary in reliance upon case distinctions: some (like Japanese) do not have them, whereas German has very strict case rules.
- Case in the titles: most words in titles started with an upper-case letter, keyword cues were not always present, and the style used was often abbreviatory. Thus, analysing these was quite different to analysing the main text. Most systems adopted the method of processing the title *after* the main text: for example, LOLITA attempted to match proper nouns in the main text to strings in the title. The author notes that such methods would not help applications which rely on processing a great number of titles alone.

In SO, no analysis was done of the relative difficulty of the MUC-6 scenario task, as was done tentatively for MUC-5 relative to MUC-4. That is, there is currently no reasonable, objective way to perform such a comparison, but relevant factors were discussed, and their quantitative effects considered. The factor of short development time in MUC-6 was seen as hindering such a comparison.

SO draws several conclusions. Extraction using "shallow" information is highly accurate, but accuracy is significantly lower when ambiguity appears, or when deeper analysis is required[2]. One possible result of the high scores in NE is to set a standard for future competitors, since the centrality of the task means it is a likely prerequisite for any IE system.

A wide range of techniques were used. From heavy use of pattern-matching and automatic learning to full-parsing and rule-based systems, with varying methods of computing output. Experience was probably significant – previous work meant that sites had "fast and efficient methods for updating their systems and monitoring their progress"[3].

The introduction of the SGML tasks has proved interesting – a new method of representing extraction results, and for many systems the possibility of new and previously unconsidered applications, especially connected to WWW browser technology.

6.2 Methodology and Infrastructure

This section discusses the underlying details of MUC-6. Such details were not considered in GSD or SO. These are how the tasks were defined, the form of the output and answer keys, consistency in the template task keys, the scoring algorithms, and the weighting of the scores. These issues are inter-related, hence the occasional forward reference.

[2] These are hardly controversial points, but do serve to confirm that no great technological breakthroughs have been achieved by any of the MUC-6 competitors.

[3] LOLITA was the only system not benefitting from such experience (SO, p. 25). Only recently have there been developed methods for detailed checking of performance, eg the template comparison tool.

6.2.1 General Task Definition

The MUC-6 specifications are informal text documents. [Sundheim, 1996] refers to them as task *explanations*. They mirror the output form of the task: the template task specifications present the rules for filling each part of all templates, and the SGML task specifications attempt to define what is markable, and for NE, how to decide on a markable's type.

Common to all tasks is a set of tokenisation rules. This document, not included in the MUC-6 proceedings, is a list of rules on how to treat punctuation and hyphens (in particular, end of line hyphens). It seems intended for the SGML tasks (defining the exact extent of a markup), but is referenced in the basic Information Extraction document. Most of these rules are obvious. The author notes that the requirement of final full stops to be included when marking an abbreviation is never enforced by the scorer, which ignores such 'post-modifiers'. The other post-modifiers are comma, semicolon, and apostrophe. This calls in to question the existence of such a rule, or the policy in scoring. In particular, why are post-modifiers (along with pre-modifiers and corporate designators) optional? In one sense, some of them are "information-free" after a system has marked the attached phrases with a certain type, hence unimportant to a user. Such optionality may be reducing the difficulty of a task.

The template task specifications are fairly straightforward. They have the advantage of being defined in terms of a fixed template structure: each feature of a template is more or less independent (the cases which are not are discussed in section 6.2.2). Reference is made to the NE task specification to define the scope of names. Note that the style of definition of template tasks has been refined over the previous MUC events.

The SGML task specifications are a series of statements which attempt to define 'markables'. Coreference links are to be added between markables, and NE markables require a type. This form of definition is harder to understand than the template style, and certainly harder to check one's comprehension without a detailed re-reading to look for relevant sections of specification. However, the tasks being described are intrinsically harder to define, and are not amenable to a template-style definition. Uncontroversial linguistic detail (such as simple phrase types) occurs in these specifications, to categorise related phenomena. Coreference depends on Named Entity, as no substring of an NE markable may coreference.

There are few points to criticise, and these are minor. The ambiguities or inconsistencies occur only in minor areas, which is confirmed by good scores and the high inter-annotator agreement scores; hence they had a limited effect overall. We offer the following observations:

- Consistency of (person) title handling (eg "Mr."). Titles are not allowed in NE markables, but are allowed in CO markables – though need not appear because of the MIN string rule. In templates, person names follow the rules for NE and such titles should appear in the separate PER_TITLE slot.
- Some of the linguistic distinctions may not be important to *real* users; they may increase the artificiality of a task.
- These specifications have a *dual* purpose: as instructions to the annotators producing the model answers, *and* as instructions to the developers of the competing systems. A system will reflect its developers' understanding of a task: it cannot perform well if it has not been programmed successfully. Thus, developer understanding is a limiting factor in performance.
- It is clear that the specifications are not the kind that are usually prepared for software systems. They are very informal, and it is not obvious whether the specifications are complete and consistent, to name two common attributes for assessment. This raises the interesting question of what form NL system specifications should take. We consider this question in chapter 8.
- Some of the difficulties in the specifications could be in trying to supply the detail necessitated by the scoring techniques. The features of an answer must be presented in a fixed way. (This issue is discussed in section 6.2.4).

It is possible that similar results (of IA scores and system scores) could be produced from looser guidelines. To the author's knowledge, no experimental work has been attempted to prove or disprove this. It seems likely that real text scanning tasks for humans are defined without such complexity of detail – but relying on the experience and knowledge of the human agent, but again, no evidence has been found in the public domain.

- Some of the specifications *restrict* what is to be output. Such restrictions could be aimed at encouraging 'deeper' processing to distinguish between valid and invalid cases of some pattern. The rationale for these restrictions is not stated explicitly. Again, we ask, are these restrictions useful to real users?
- Finally, there is the question of the status of the specifications. Are they requirements of what a system must do, or just loose guidelines to expected behaviour? It could be an interesting experiment to test the adherence of MUC-6 competitors to the task definitions.

To summarise, it is hard to define an NL task. There are particular problems of consistency and completeness. Definition is easier when the output follows some framework, eg templates. Detail seems to be required by the limitations of the scoring methods. Even then, the MUC-6 specifications rely on human interpretation: they could not be executed by a machine in any sense.

6.2.2 The Form of the Answers

SGML Style

The SGML tasks are to identify certain entities in the text and annotate them with some information. The SGML style is well suited for NE output because it can highlight certain *independent* strings in the text (and the results may be easily visualised using a WWW browser), but there are reservations about its suitability or necessity for CO output. In particular, we note that the output format suitable for scoring may not be suitable for visualisation.

The SGML is not naturally readable, and is usually visualised by some other means. The SRA "Discourse Tagging Tool" GUI can display links in an article by lines connecting markups. Colour-coding in a WWW browser can also be used to differentiate chains. This conversion means that little would be lost if coref results were required as templates containing start and end offsets.

Because the SGML format is so compact, viewing differences between output (eg between key and response, or versions of response) is difficult. Variations in markup identifiers can complicate this. The visualisation methods above do not help. The template comparison tool can provide such displays, albeit in template form, which the author has found very useful. An overall view of Coref is possible by using the comparison tool to produce a daVinci-showable graph: this shows the alignment of key and response chains, and makes omissions and chain intersections particularly clear.

Template Style

There are two issues: the issue of representing information as structures containing slots of simple information (eg strings or values from a small set) and of links to other such structures, and the issue of the way these structures are defined – ie, the different types and their content. The first issue is hardly controversial – this method is suitable for any information, and is system-independent, although there are problems of how to represent alternatives. We shall consider the second issue.

[Onyshkevych, 1993] discusses template design: "The design of the template for an information extraction application (or exercise) reflects the nature of the task and therefore crucially affects the success of the attempt to capture information from the text." He then discusses some of the attributes to be considered:

- Descriptive Adequacy: eg units must be given where quantities appear.

- Clarity: the information should be in an unambiguous and final form, not needing large amounts of additional inference.
- Determinacy: there is only one way of representing a fact.
- Perspicuity: the template should be conceptually understandable to human viewers, if humans need to view the templates.
- Monotonicity: the templates incrementally represent the information, so that further templates do not refute earlier information.
- Application Considerations: eg complexity of linkage – in the context of MUCs, whether the templates can be feasibly scored (ie, within reasonable time and space limits).
- Reusability: the template design is useful across domains.

Another issue considered is of 'flat' vs. "object oriented" templates. The former were used in MUC-3 and MUC-4, with 24-slot templates. Representing information in these was found hard, and there were often many blank slots. In contrast, MUC-5 used 11 types of template (in the Joint Ventures task, fewer in the Microelectronics task). The template design was developed by "reconciliation of multiple, often contradictory goals", and were "(implicitly) designed to cover a range of linguistic phenomena (eg coreference, metonymy, implicature) and to (explicitly) require [sic] the full range of Information Extraction techniques (eg string fills, normalisation, small-set classification, large-set classification)". Pragmatic programming considerations also gave a need for reducing the design complexity. Unlike MUC-6 where the templates form a directed acyclic graph, the templates in MUC-5 can form a *cyclic* graph – though most of the types were acyclically related[4]. Onyshkevych admits that monotonicity was not maintained in some areas of the design, introducing a bias in scoring (ie, provided more scorable features without increasing information content).

One innovation in MUC-6 was to formalise the general structure of scenario templates into layers. At the bottom are the entities participating in the story. A separate task tests extraction of such entities from a text. The relevant events of the story are represented by templates in the top layer, which connect to their entities by the middle layer of relationship templates. An additional template collects together the relevant event templates. This structure may have been useful in implementing the task too, as the design provided a framework which could be instantiated with scenario-specific rules.

We raise the following points on general template design:

- Most significantly, the literature seen implies that a task must use a *single*, all-purpose template design. We reject this, on the grounds that the requirements are contradictory, as now explained.
- Perspicuity implies that the template design must be easily understood by humans if they are expected to view the results. This conflicts with the monotonicity and clarity requirements. A typical presentation of non-trivial material, such as a newspaper article, is in layers: initial sentences set the scene, and successive ones add and refine detail. The whole meaning of the article is not presented in one large, but logically correct, piece.
- Furthermore, different applications may require different *aspects* of the information. Should the representation attempt to satisfy all possible future uses? Alternatively, a basic representation of the underlying information could allow simple derivation of the required 'view' as required. If the algorithm is a bijective mapping – ie, one which may be reversed without

[4]The author is uncertain on whether back links are scored or not. The task specifications were not published in the MUC-5 proceedings (nor in the related TIPSTER volume). Furthermore, the graph on page 8 of the MUC-5 proceedings does not seem consistent with score tables in the back. Varying POS counts in the score tables suggest that some of the templates with possible back links were optional. Scoring of back links would introduce a bias if the opposite link was also scored, since that information had already been scored. This assumes that producing a back link is trivial from a programming point of view.

need for search or heuristics, and hence its use need not be evaluated – then evaluation need not consider the human-oriented views, or need to base scoring methods on them: the underlying representation could be scored. This question needs further research.

- Determinacy would be satisfied by a basic representation of the information, if all information was represented once only and in a simple form.

- The issue of the 'soundness' of the resulting template language has not been considered. Some designs may allow representation of nonsense, such as having many slots in a template which can be filled in a contradictory way (although in practice most should be left empty), or in the Scenario task where the SUCCESSION_ORG, OTHER_ORG and REL_OTHER_ORG slots can be used to express that an organisation is not related to itself. The ST design also allows duplication of this pattern through another IN_AND_OUT (ie, a bias in scoring), which then allows one to assert the contrary. A stronger design would limit what could be expressed, and prevent systems making some mistakes. Monotonicity would be helped as the language would be more restricted.

- The scoring method forces one template per entity, which means systems must identify separate occurrences of an entity and then successfully merge the relevant information from separate occurrences. This requirement is not explicitly stated in the design: our point is asking whether it is a deliberate or desired part of a task. It has a non-trivial effect on scoring.

  Consider: "Shakespeare wrote the play 'Titus Andronicus'. The beard of the author of 'Macbeth' is black." Systems that lacked the relevant background knowledge may still be able to process the information in each sentence separately. But in the current framework, they would either lose marks by omitting information (eg discarding the information from either or both sentences) or by over-generating (eg producing two templates containing the separate information). Credit is given for the correct separate analyses *if and only if* the *overall* analysis is correct.

  This point does constitute extra difficulty in a task, although most cases (in newspaper articles) will be determinable on the basis of similarity of names or the simplicity of anaphora, and hence will be relatively trivial.

  One possible method of making such associations scorable is to have a template for each relevant mention (ie, yielding relevant information) of an entity in a text – which will be a string taken from the article, with 'identity' links between the templates. This is a merging of ideas from the template and coreference tasks. Another possibility is to allow several incomplete templates instead of one complete template, and to require the *scorer* to align fragments with a key template.

- The 'completeness' is also important, but harder to assess as it would require considering possible circumstances of a story. Optionality or ambiguity is hard to represent in the Scenario task, especially as templates cannot be marked as optional in this task (they can in TE). An example is article 9306240111. Here, optionality is represented via alternative link sets in the CONTENT slot; six links occur in both sets, with the seventh included in one set only. Adding parentheses to the language would simplify this construct (ie, `A B C D E F (G / )`). The choice of which alternative link set should depend on the interpretation of an organisation referenced in the 'optional' SUCCESSION_EVENT. The author cannot find evidence that the scorer enforces this distant dependency.

  Furthermore, is the language complete enough to represent incomplete analyses by a system? A system cannot receive credit for correctly analysing part of the text if the recovered information cannot be validly expressed. This is another way of stating the previous point.

  Considering deeper points of task design, should we expect a system to handle a subtly ambiguous text? It may be better to expect the system to perform the simpler parts of a task, and highlight ambiguous situations if they are relevant to the task. After all, humans are currently much better at disambiguating.

- The template design seems oriented towards static information, ie summarising a state of affairs. Temporal information is hard to represent. Consider a management succession which records two job changes of one person in a short time. There is a possible implicit assumption that situations are rarely that complex – so the template language in MUC-6 does not need the extra flexibility.

- The relationships and dependencies between template types are also an important part of the design. Part of the relationship is enforced by the actual fill rules. A particular case is of *necessary slots*. The existence of instances of some template types depend on the filling of certain slots, such as entity name. This seems to restrict the tasks to use entities which are clearly identifiable. Additionally, it penalises systems which have problems with the required slots even if they are correct with the other information (LOLITA suffered this a few times).

  A consequence of this is when a template's existence depends on the existence of another, such as a SUCCESSION_EVENT depending on obtaining a fill for SUCCESSION_ORG, hence on finding a legal ORGANISATION (ie, one with a name or a description). It is conceivable that a SUCCESSION_EVENT (and many points) could be lost if a mistake is made in the target of the SUCCESSION_ORG. Although this does require a system to filter out job changes that do not mention a company, it is not certain that this was intended by the organisers: such dependencies and their consequences are not well justified. The constraints may also be providing easy marks, if combined with typical features of the input.

- There is no way to indicate system uncertainty in responses. It is feasible that a system may have an incomplete template, which it may discard and maybe lose recall points, or may produce it with guesses for the missing slots and risk loss of precision. There is no compromise position. One solution is to reduce the detail in templates, ie the granularity.

From these points, we conclude, we reject the idea of single, general-purpose template design in favour of a simpler representation of the information required, from which human-usable views of the information can be easily derived. At least two sites disliked the MUC-6 template design and one of these analysed text into their *own* model of succession events, post-processing to produce the MUC-6 form.

In answer to the problems of coreferencing entities and representing related information, we suggest discarding the one entity-one template idea in favour of templates which makes more explicit the facts required to build an answer. This means that credit can be given for the implicit steps and that system weaknesses in specific areas do not penalise stronger areas through loss of a 'keystone' fact.

Scoring may be a problem for such reduced templates (see section 6.2.4). However, a constraint of basing each template instance on a specific string from the input text should reduce alignment ambiguity. It can also eliminate the alignment problems we saw in section 4.5, such as confusing identical job titles (eg when an article mentions the presidents of several companies), and it has the advantage of providing a very strong link from the extracted information to the associated text. No such link is possible in the current MUC-6 template scheme, and it mirrors the association seen in the SGML tasks. A point-and-click style GUI, such as CRL's "Tabula Rasa" [CRL, 1997], could help to produce such templates. It may even be possible to encode template output as SGML, using markup attributes to encode slot information.

The problem remains of delimiting the information to extract, since the lower granularity of detail means removing the higher level constraints on content. For example, much irrelevant or useless information could be included about minor figures in a scenario event. One constraint will come from the relevancy conditions of the task (what the user wants from the article). A second constraint will come from the form the user requires it in – the higher level views.

In summary, we have suggested several changes in template format. The most important of these is basing each template instance on one or more specific strings from the input, which should reduce alignment ambiguity and provide a stronger link from extracted information to original text. The second most important suggestion is discarding the idea of an all-purpose template

language: a simple language which can represent all detail at a low granularity should be the scorable output, with human-readable views of this information automatically derivable from it. Finally, we observe that template design is a special case of designing knowledge representations. The general problem is unsolved, so we should not assume an easy solution can be found for sub-problems.

6.2.3 Template Task Key Consistency

The two template tasks share the definition of PERSON and ORGANIZATION templates. TE requires templates for all such valid entities, whereas ST requires a subset: only those relevant to the succession events. Thus, we expect the PERSON and ORGANIZATION templates in ST keys to be (content-) identical to their counterparts in TE keys.

After noticing two inconsistencies in the keys for the article of chapter 4, the author tested this using the template comparison tool (Appendix B), using the template scorer with modified configuration files (essentially a union of those for TE and ST) to produce the scorer map histories needed by the tool.

Only a few errors were discovered: the two errors in one article was atypical. In addition to finding actual discrepancies (listed below), the checking process also uncovered the following bug in handling of PER_ALIAS and ORG_ALIAS slots in the version of the scorer we were using[5]. The following template scored against itself produces an f-score of only 66.67. Placing both alias strings on the same line produces the expected score of 100.00.

```
<PERSON-123-1> :=
    PER_ALIAS: "a"
               "b"
    PER_NAME:  "fred"
```

The content discrepancies are:

- TE template PERSON-9305040023-5 contains a a misuse of double quotes - the ST key uses single quotes.
- Optional TE ORGANIZATION 9312030175-3 includes an ORG_LOCALE and ORG_COUNTRY which are missing in the ST key.
- (ST) ORGANIZATION 9306220057-1 (number 9306220057-2 in TE) has an extra ORG_DESCRIPTOR alternative. Or rather, the TE key is missing a valid alternative.
- (ST) ORGANIZATION-2 in the same article omits the ORG_DESCRIPTOR 'subsidiary'.
- ORGANIZATION 9311150068-1 has the LOCALE type 'CITY' in TE, but 'PROVINCE' in ST. The text does not indicate which is correct.
- ORGANIZATION 9403160006-2 is missing an ORG_ALIAS "Paramount Pictures" in the ST key.

We conclude that the consistency was not checked by the MUC-6 organisers. It is a weakness in the methodology not to do so when task definitions are shared. Fortunately, the difference is small in the subset used in ST: six errors among the 120 ORGANIZATION templates and one among the 137 PERSON templates (TE has 606 ORGANIZATIONs and 496 PERSONs).

[5] We are using C scorer v1.4: we've been told that this bug has been corrected in later versions of the scorer.

6.2.4 Algorithms for Scoring

This section discusses the methods used to score a system's response in a task. It considers the steps of alignment and score counting for the template tasks, and then the specific techniques for scoring coreference. Note that the SGML tasks are converted to pseudo-templates and scored in a similar manner to proper templates (with the exception of checking text offsets and the counting specific to coref), so the template details are relevant to all tasks.

General points

A certain amount of preprocessing is done. Pre-modifiers (`A AN THE`) and post-modifiers (`; ' .`) are removed, then various corporate designators such as 'company', 'no liability', etc. As noted above, some of these contradict the spirit of the tokenisation rules.

The answer keys are specified in a precise way, with a small degree of freedom provided by the ability to make some features optional. Scoring proceeds by determining which part of the key should correspond to which part of the response ('alignment') and then counting the agreements and disagreements with the key.

In situations of ambiguity in scoring, the mechanisms seem to prefer matches leading to a larger score than to an accurate match. The inconvenience of this policy to diagnostic interests has already been mentioned (eg, in section 4.5).

Template Scoring

A first comment is that alignment, or mapping, is a hard process. It must establish one-one links between templates in the key and the response. The process is a compromise between accuracy, time, and effect on scores. An exhaustive test of the overall scores of all article-wide mappings would be prohibitively expensive, and not guaranteed to return the 'correct' match – especially when the responses differ widely from the keys. Alignment is effectively an unsolved problem [Chinchor and Dungca, 1995].

The algorithm used in the MUC-6 scorers creates all possible pairings of templates of a given type: the slots in each pair are scored and weighted by an amount specified in the configuration file for the particular slot type. The overall weighted sum is compared to a similarly specified template threshold, and the pair accepted if it is above threshold. The accepted pairs are scored conventionally and sorted by decreasing f-score. The highest matches between any two unmapped templates are then accepted as final. The hierarchical template structure of ST is mapped bottom-up, as templates include links to lower levels: links to templates in the response must be resolved to their key equivalents. Additionally, there is an *undocumented* special treatment for IN_AND_OUT templates: the key can legally have several instances of IN_AND_OUT templates which are identical in content, and only distinguished by their parents. Mis-mapping is a real danger, so each instance contains links to identical instances which can be used to determine the optimal instance mapping for each parent.

The weighting system is NOT used in MUC-6 (apart from requiring at least one slot match in a template for a hypothesised match). This was intentional: after experimenting with different weights and thresholds, the MUC organisers had decided ([Chinchor and Dungca, 1995] p35) that no particular slot should define an object. In particular, there was a difficulty in reaching threshold in sparsely filled templates [Sundheim, 1995].

Warning: the C scorers represent the initial f-score as a fraction of 100.0, leading to weighted sums of several hundred. The thresholds in the distributed configuration files are all units, which need to be changed to hundreds if weighting is required. This conflicts with the explanation in the `emacs` scorer manual [Chinchor, 1995a].

We raise the following points:

- [Chinchor and Dungca, 1995] admits that the process does not attempt to minimise the number of unmatched templates. For example, a semantically wrong match can score

more than a poor but correct match, allowing the possibility of one key template and one response template with no scoring matches, hence leaving them unmatched. The effect of this is reduced slightly by the additional policy of pairing each unmatched key template to an unmatched response template during the counting stage of scoring. This converts a joint MIS (for under-generation) and OVG (over-generation) penalty to a smaller INC penalty. Both techniques are unhelpful to diagnostic studies.

- The choice of statistic for sorting is debatable. Identical f-scores can be obtained from matches with different weighted sums, and the C sorting routine used (`qsort`) does not guarantee a predictable order for equal elements. There is a strong intuitive argument for preferring matches with higher weighted sums (assuming use of positive weights). Note that this only applies when weighting is used.

- The scoring method assumes a one-one match is reasonable. This requires a system to produce the same number or fewer templates than the key to make sense: it has problems when the system over-generates. We argued against the one-one match on p. 117, observing that it requires successful integration of all information about an entity.

- That the scheme for handling IN_AND_OUT ambiguity only applies to the key templates is more evidence of the unstated assumption that a system's response must be fairly similar to the key. Such a scheme would be useful for responses: it is possible for a system to produce content-identical templates that have no obvious match in the key, especially if the important parts of the template are missing or wrong.

- The mapping process can be improved by associating each text string from an article with its position in the text, as is done effectively for the SGML tasks. This would allow use of templates with a small number of slots, which normally are hard to align. There is no reason why a template based on a single string cannot be encoded as an SGML markup with a set of attributes.

Scoring Coreference

Coreference SGML is converted to templates and aligned as templates, except for the generous overlap conditions through which MIN strings are implemented (anything between the full markup and MIN string is accepted). The scoring technique [Vilain *et al.*, 1995] is reasonable and clear. One may ask whether additional credit should be given for producing markables which do not contribute to the score. Possibilities include being linked to the wrong chain. For example, markup B3 when the key includes chains {K1, K2} and {K3, K4, K5} and the response includes {B1, B3} and {B4, B5} (equivalence of number means alignment). Credit could also be given for producing markable B1, even if no B2 is produced, as B1 only appears because of the error of B3 (without B3, that response chain would contain only one markup and hence not be a valid coreference). This credit for producing mappable markables could be quoted as a supplement to the linkage score.

6.2.5 Final Score Weighting

This section considers how the detail of scorer output should be converted to a smaller set of numbers, potentially two ratios (recall and precision), or one (weighted) combination of ratios (the f-score). This reduction is not necessary, but it is easier to understand than a large set of numbers. Currently, all scorable features are worth one point, and the overall scores are summed from individual features in all articles before the ratios are calculated. Note that smaller articles will usually contribute fewer features and hence will contribute less to the final score than larger articles. The question is, is this lack of weighting acceptable? The answer should depend on the purpose for running the evaluation.

Intuitively, some features of an answer are more important than others in applications, sometimes to the extent that mistakes on certain features are unacceptable. Therefore, if the purpose is to test suitability for a non-trivial application, then the uniform weighting seems unacceptable.

On the other hand, *how* to weight things is not obvious: perhaps the most reasonable way is to relate features to their value in a realistic application, ie how much the "added value" of the extracted information is worth and what the consequences of error are.

MUC-6 could instead be intended to discriminate between competing systems and determine the 'best', where 'best' is defined as getting the highest number of features correct without making too many mistakes. If only higher numbers are important, then the less weighting the better: good marks can be obtained through getting the simpler features correct with straightforward techniques, and extra marks by conservatively attempting the harder features.

A third view is that easy-to-get features should be less valuable than harder ones. Annotators could mark some indication of difficulty – such as which 'level' of analysis is required to get the correct answer. Of course, there are problems of definition and of agreement over what characterises possible ways of analysing language, and logistic ones of the extra effort of annotation. If one makes the assumption that the 'easiness' of an answer is indicated by the number of systems getting it correct, then the extra work can be done automatically.

The author has implemented and analysed this basic idea in section 5.6, producing graphs for each task. How do we reduce the graphed results to a simpler value? The obvious suggestion is to normalise the score in each correctness class by the order of the class or by the class's size, and take the average over the classes. Thus we have the weightings corresponding to a result being twice as important as another if half as many systems get it, or being twice as important as another if there are half as many instances in the respective classes. The latter weighting takes into account the noted differences in distribution of class size, which we interpret as indicating task difficulty (ie, is there a large number of features which most systems get correct?). The author has not performed these calculations: the above is a suggestion of one application of the method of section 5.6.

In summary, there would appear to be no universally correct answer to the weighting question, especially as MUC-6 does not state its goals with respect to this issue, so one point per feature would seem most reasonable for a summary of results. We would expect something more sophisticated in a real IE application.

6.3 The Specific Tasks

This section contains comments on the individual tasks. As the major points on the MUC-6 methodology have now been discussed, this section will be brief. We can only criticise small details *within* the tasks. We include the conclusions from section 5.6, on "Correctness Analysis". The question of whether task designs meet their goals is considered in the chapter conclusions. We have little grounds for criticising task choice: that is more a matter for the organisers, of whether the evaluation yielded the information they required.

6.3.1 Named Entity Task

Task Definition

The document contains an overview with markup-type specific 'guidelines' covering the main phenomena to be marked. These guidelines are not a complete set of precise statements, and seem more oriented towards annotators. Some of the guidelines refer to case distinctions. Some classes of proper name were excluded from the markable set. The document depends on the tokenisation rule document and the information extraction task document (for definition of an alias expression). The latter itself refers to the NE document for the definition of a markable person (name).

How systems did.

Most systems did very well, with eleven of the twenty entries obtaining f-measures over 90%, which compared well with human (experienced annotator) performance.

Discussion

SO points out several limitations and features of this task. Briefly, they are:

- Restriction of corpus to journalistic writing, biased towards the subject of the ST task.
- The small size of the corpus.
- Accurate usage of case, which provides strong cues to markability.
- 82% ENAMEX markups to 8% NUMEX and 10% TIMEX. Inside ENAMEX, ORGANIZATIONs formed 48% to PERSONs 40% and LOCATIONs 12%. Hence, person and organisation names dominated the scorable features. Sometimes it was difficult to determine the type of a name. Person names generally drew a smaller error rate than locations, and locations than organisations. A possible reason is the greater variety of organisation names over person names.
- SO draws attention to the possible complexity of some of the names, but does not provide evidence of how common these complex names were, or that any system was in fact getting these correct. There are several such anecdotes in SO.
- For most systems, the headlines were analysed with more error than the text body. Most systems analysed the headline after the body.
- SO's particular criticism of the task is its limited scope – a variety of proper names were excluded.

Most of our criticisms are covered by these points. Together, they suggest that systems able to use simple cues in proper names will do well, and that the text provides many such cues reliably. Correctness analysis (figure 5.6) confirms this, showing very many features on which almost all systems are correct. Scores are very similar in this region, but a great variability of scores in the region where less than half of the systems are correct.

There appears to be an anomaly in scoring: it is possible for the type to be correct if the text is wrong. LOLITA's scores for article 9301190098 show COR for type exceeding that for text. Since NE requires an exact match of markable string, allowing a type to be correct where text is wrong is counter-intuitive. Reading from the scorer source code `libmap/map.c`, it appears that a type match without an exact text match is allowed if the key and response markups overlap at *any* point. Another example appears in the NE analysis for LOCATIONs on page 45.

6.3.2 Coreference Task

Task Definition

The task definition attempts to characterise markables and to indicate when a coreference relation exists between markables. Substrings of Named Entity markables are not markable. The definition is not direct, in the sense that after defining the basic task, certain cases are then declared unmarkable. With such an approach, it is not easy to check consistency of the definition. Systems may mark anything between the full key string of the markup and the MIN string recorded in the tag attributes. We notice that (person) titles are handled differently in the SGML tasks: they are *not* considered part of an NE markable, whereas they are in CO. The NE treatment is in line with the Template Element task, where PERSON templates contain a separate PER_TITLE slot.

One problem in the definition noticed by the author is in section 5.4, on Time-dependent Identity: "Two markables should be recorded as coreferential if the text asserts them to be

coreferential at ANY TIME." (upper case in original). This allows coreference between a person and the two jobs that he is moving between, but may cause a problem when another person is moving into the vacated job. Note that the scoring method cannot handle a string which is marked twice[6]. Marking both persons as coreferential is clearly nonsense. Thus, we conclude that this portion of the task definition needs revision.

How systems did

These points are taken from Sundheim's overview. Seven systems competed, most scoring around 50-60% recall and 60-70% precision. About half of the competitors concentrated on individual coref, which had benefits for other tasks. That is, the coreferences involving people. Which systems did is not detectable from the scores. Several sites estimate that good name/alias recognition provided around 30% recall and 90% precision. In the walk-through article, some systems scored much better than the overall score, mainly because of the unusually high number of personal pronouns and names. Inter-annotator agreement was low for this task.

Discussion

The limitations of this task are similar to the ones for NE above. As noted by SO, the expense of annotation restricted the detail of scoring. Several methods were planned, including the annotation of each markable with its basic grammatical type and its semantic type. Work along these lines is under way by Urbanowicz[7] at Durham.

There is an error in the MIN strings in some of the keys: the annotator has attempted to represent alternatives with a | inside a markup's MIN string. This is not sanctioned by the task definition. The affected articles and markups are: 930119-0098 IDs 20 and 45, 930504-0023 ID 35, 940127-0105 ID 125, and 940420-0037 ID 11.

Coref poses some problems for correctness analysis in its current form. A tentative result appears in figure 5.7: it shows a smoother distribution of difficulty than NE.

6.3.3 Template Elements

Task Definition

Template task definitions are relatively straightforward as they follow a template structure and provide rules for filling the templates, sometimes with conditions on the well-formedness of the template.

The template design highlights the importance of entity *type* in information extraction. It also increases the penalty of getting it wrong. ORGANIZATIONs represent a bigger challenge as there are more slots to fill than for PERSONs.

How systems did

Most systems scored f-measures in the 70-80% range. PERSON templates scored better than ORGANIZATIONs, to be expected as they are simpler. An analysis of error rates per slot for ORGANIZATIONs supports the idea that it is harder to fill slots which require non-trivial analysis. The above points were taken from Sundheim's overview. Note that the error analysis did ignore contribution to the final scores, so a good PERSON template is worth the same as a moderately well filled ORGANISATION, and the high error rate on the relatively infrequent ORG_DESCRIPTOR slot is not too serious.

[6] The markups would have identical content and offsets, which means mis-mapping could occur. Ie, the alignment would be unpredictable.

[7] email A.J.Urbanowicz@durham.ac.uk for further details.

Discussion

Again, the corpus was oriented towards management succession stories, having the effect of producing more entities to describe in templates. Correctness analysis (figure 5.8) shows a similar pattern to NE, but with a more gradual transition from the easy to the hard features.

6.3.4 Scenario Template

Task Definition

Again, the task definition follows template structure. ST templates are layered; the task definition proceeds top-down. Note that the formalisation of template structure was an innovation in MUC-6. Some templates depend on the filling of sub-templates for existence, leading to some repetition of fill conditions in the higher template. Sometimes, understanding the template result required knowledge of the task rules. In particular, the OTHER_ORG slot must be interpreted in conjunction with the NEW_STATUS and ON_THE_JOB slots.

How systems did

Most systems scored between 30 and 50% recall, 55-75% precision. LOLITA was the lowest scorer. SO suggests that the scores indicate improvement relative to MUC-5, as no decrease in highest f-measure was seen despite the shorter development time. Note that all sites entering ST also entered the TE and NE tasks. Furthermore, all sites except Durham had MUC-5 experience, according to SO.

Discussion

SO mentions the following points. The template design was possibly too ambitious, trying to include some peripheral facts about events which were difficult to specify and/or were not clearly reported in the articles. Certain slots which recorded this peripheral information caused problems for the annotators, especially VACANCY_REASON and ON_THE_JOB, which were defined by a set of heuristics. Future MUCs may use a simplified version of the template design, omitting these slots and omitting the IN_AND_OUT templates.

To this we add a criticism of the template language. The job changes in the scenario used are essentially dynamic, whereas templates are more suitable for static description. Hence, attempting to encode the *temporal* nature of job changes in a template design may be unwieldy. To give a (pathological) example, a person could leave a job for a second job, then return to the first job. Representing this would be difficult in the current design. The additional difficulties of representing alternatives across the templates has been mentioned before.

Template alignment was problematic in ST, especially when a system was not performing well. Errors in templates meant that they could be confused with other erroneous templates. Another problem was the repetition of common job titles (eg, talking about the presidents of several companies in the same article), which added to the confusion. The problems of interpreting templates are illustrated by the number of annotator comments explaining a template in the keys.

Correctness analysis (figure 5.9) shows a new pattern: the size of correctness class increases from easy to hard, and the f-scores follow the rise for the first few classes, and then drop. Our interpretation is that ST is harder than NE or TE, with few features which are easily obtained by the majority.

6.4 Observations and Suggestions

This section contains some ideas which may help to improve future MUCs. These are additional to the ones appearing in the previous sections. None of these have been fully implemented, mainly because of resource limits.

6.4.1 Pure Scenario Template Performance

The ST task contains TE, in that some of its output is expected in the TE output. If one system is assumed to use the same mechanism for producing ORGANIZATION and PERSON templates in these tasks, then in a sense, scoring them in the ST task is redundant. Furthermore, it is not unreasonable that performance on these parts of ST may be providing easy marks: correctness analysis showed TE to contain many features which most systems got correct. Hence, their presence may boost ST scores, without adding significantly to the performance element of detecting job changes.

So, it appears interesting to filter out the TE elements from the ST scores. We note that the PERSON and ORGANIZATION templates are required to align references in the key with those in the response, but we need not count them after alignment. The table below was produced by summing the slot totals in the score summaries for TEMPLATE, SUCCESSION_EVENT, and IN_AND_OUT templates. It lists the original recall and precision, and the same after filtering, plus the changes in recall and in precision. The systems are sorted by decreasing original f-score. As expected, the adjusted scores are slightly worse. Note that the drop in precision is more than the drop in recall in all cases.

	Original		Filtered		Difference	
System	Rec	Pre	Rec	Pre	δRec	δPre
NYU	47	70	46	66	-1	-4
BBN	50	59	50	56	0	-3
SRA_base	47	62	44	57	-3	-5
LockheedMartin	43	64	40	59	-3	-5
SRA_recall	58	46	56	42	-2	-4
SRI	44	61	42	58	-2	-3
Sheffield	37	73	35	70	-2	-3
U. Man.	39	62	38	57	-1	-5
SRA_precision	32	66	30	61	-2	-5
U. Mass.	36	46	35	42	-1	-4
Durham	33	34	31	30	-2	-4

What can we conclude? Our point is that this method is another way of analysing ST performance, independently from TE performance. As such, it is a different way of deriving a score. Hence, nothing can be concluded from the difference: the table is presented as evidence that a difference exists, whether statistically significant or not. The important decision is whether one accepts the argument of task overlap, and agrees that this altered score is an interesting view on ST performance.

A more interesting experiment will be to examine the difference to the correctness analysis. We expect a difference in the easier part of the graph.

6.4.2 Defining a Baseline

It may be argued that the easier parts are so simple as to be meaningless in a thorough evaluation. For example, a very simple pattern matching program could get them correct. The number of easy features can obscure the detail of performance on the less easy answers – which may be more interesting in evaluating performance. One may want to show how a supposedly more complex system does better than a simple pattern matcher. Of course, a system should perform well on the easy parts, so we cannot ignore this component completely.

One way of defining this baseline is by implementing a collection of basic rules in a program, specifically those which are successful on the easier parts of a task. An example rule is, for Named Entity, a series of capitalised words followed by a corporate designator is an organisation. Another rule is the more general one linking repeated identical names. However, there is a problem of objectively selecting representative rules.

Section 6.2.5 discussed weighting the parts of an answer by how easy or difficult they were. The measure of easiness suggested was based on the correctness analyses of section 5.6. The correctness analysis can also be used to eliminate the easier parts of a task by how well systems performed. This is more principled than any ad hoc attempt to characterise difficulty. We can partition the keys for a task at some threshold point, and score each part separately. Separate recall and precision figures can then be quoted for the (easier) below-threshold and (harder) above-threshold sections. Good systems should have similar high scores for the below-threshold portion, with differences seen above the threshold. Other systems will have weaker below-threshold scores, but their above-threshold scores may be comparable with the good systems.

How do we choose a threshold? The possibilities include:

- Number of correctness classes: for example, take the top m classes (from $C(m-n)$ to Cn) as below-threshold, and the remainder above. Choice of m is arbitrary, so we could choose some percentage of n (eg 50%).
- Percentage of overall answer: we rank the features by correctness, and take the easier x% as below threshold, the remainder above.

There is no reason why we must have a single number. Since choice of the cut-off point is arbitrary, we could graph the resulting recall and precision against the cutoff point, and compare graphs for each system.

6.4.3 Assisting Annotation

[Sundheim, 1996] mentions the method used in the scoring of results in the later TREC events. (TREC is a sister event of MUC, concerned with information retrieval.) The volume of articles used in IR evaluations ("tens of thousands") prohibits human marking of relevance, so the following is used. IR systems typically produce a measure of relevance, leading to a relevance ranking of all documents given some query. The highest-ranked articles from all competitors are pooled, for example the top 200 in TREC-3, and their relevance judged by hand. Sundheim says this method "can be fairly certain to result in a reasonably complete list of relevant documents (perhaps over 80% complete, on average across queries)". This method therefore uses the competing systems to *suggest* which articles should be checked manually. Note that it does not produce a perfect answer, but one that is acceptable given the alternative of manually judging the test corpus.

Can this idea be applied to IE, or to NLP in general, of using the systems to be evaluated to reduce the amount of work (and resources) involved in producing answers? Automatic construction of most of the answer would be most useful, but a lesser form of providing a skeleton answer with identified controversial points, which a human analyst could resolve with the aid of this automatically produced information, could also save work. Sundheim[8] says that the standard pooling method would not help MUC tasks because, at the level of slot fills (or markups), there is "an indefinite number (and, to some extent, an indefinite variety) of scorable items per document". Sundheim also points out that the pooling method does not give fully correct answers.

However, the author believes that these objections are not sufficient to reject the idea. We observe, when a number of systems have been produced for some task:

- The systems do embody some knowledge about the task.
- The systems will be right some of the time.

[8] Personal communication.

- Most systems will frequently agree on the simpler answers.

Thus, competing systems have some use. The problem now exists of how to utilise their performance, or how to combine their output into something which contains some of the required answer, or could help an analyst quickly produce the answer. In their MUC-6 paper [Krupka, 1995], SRA discuss the problem of combining a high precision, low recall system with a low precision, high recall system with the aim of producing a better system; they found this problem non-trivial, and did not solve it.

Alignment of the output of different systems is one component of our problem. It may be helped by adopting the earlier suggestion of basing templates on specific strings from the input text. This could also help human arbiters decide quickly and easily when needed, as they can directly access the relevant portion of the text. Assuming we can align reasonably well, the next problem is combining information about the entities represented by the specific strings. We suggest an algorithm based on consensus: the more systems that suggest a feature of the answer, the more likely it is to be correct. Some form of weighting of evidence could be used – a system that agrees with others at many points is likely to be more informative than one that agrees rarely. A threshold could be employed, so controversial features which do not exceed the threshold are flagged as needing human judgement.

There will of course be some features which all systems miss, and some false positives – mistakes that several systems make. As Sundheim noted, we cannot be sure of good accuracy. We could gauge accuracy by manually checking some of the automatic results. Another possibility is a philosophical shift: we adopt the notion of *relative correctness*. Sundheim notes (in [Sundheim, 1996]) that the notion of perfect truth is problematic in IE work, with factors such as human mis-understanding of task requirements and the inherent ambiguity of text. A relative measure will answer the question: "from a set of systems, which performs the best relative to the others?" This contrasts against the *absolute* measure used in MUC-6. If one's reason for evaluating is to find this 'best' system for a task, then relative correctness may be sufficient. Alternately, what kind of result is really required in evaluation?

We may still need some idea of absolute correctness, to determine whether the best system will perform well enough to be profitably used. For example, it (and the other systems) could still be missing important parts of an answer. Such measures need more research: the MUC-6 measures do not qualify, as they do not take account of the importance of information, for example.

A further problem is with trainable systems. They need a supply of annotated data to train on, both for external results (eg task output) and often for their internal components (eg parsing). Our suggestions here do not offer much help. We note that provision of data for trainable systems is a problem faced by the NLP field in general: the MUC-6 system papers contained complaints that not enough data was provided.

In summary, the pooling idea used in TREC-3 may be useful in some form to make use of the agreement between competing systems. We expect it could speed up annotation by helping with the simpler parts of a task, and by highlighting controversial parts. Some change to the task output may be required, eg use of specific strings in templates, which allow quick access to the relevant part of the input. The results would not be perfect, and some human consistency checking may be required. The final result would be relative: indicating which of the competitors was most correct. This may need checking with absolute measures, to determine if the best is good enough for an envisaged application.

This scheme is still hypothetical, and there are several problems to solve, but experience with correctness analysis (section 5.6) suggests that the idea deserves further research.

6.4.4 Text Rearrangements

Providing texts and task keys is expensive, so in this section we investigate what additional use may be made of them. We discuss simple, automatable changes to the material (the text, and the corresponding portions of the key) that preserve meaning but will alter the difficulty of analysing the texts. Some of these will be general, and others will rely on the information about the text

that is represented in task keys. The results will illustrate the dependence of systems on certain cues, and their robustness and reliability when such cues are removed.

The transformations will have the effect of eliminating the simple parts of a task by attacking what makes them simple. This is in contrast to our earlier suggestion of discarding them from the scores; thus, we do not reduce the amount of testable material. Another view is that different areas of a system will be tested. For example, more use of inference with indirect cues may be required when we remove the direct cues.

In the context of MUC-6, systems which perform well on the transformed texts are not necessarily the best. As more transformations are applied, the tasks will become more artificial and removed from ordinary use. Instead, we see this exercise as "information gathering".

Transformations can be combined to produce harder tests. There is obviously some limit to how many transformations can be applied before the text becomes incomprehensible (and hence the evaluation results become meaningless). There may be problems in understanding the meaning of complex combinations of transformation. We note that resource limits, particularly machine time, will provide a practical constraint on how many transformation combinations are tried. We do not envisage more than three kinds of transformations being used at once.

A related issue is how widely the transformations are applied. Universal application may produce a bad text. Testing single changes will be too resource-hungry, and may not show interesting changes in score. Some standard amounts like successive quarters (25%, 50%, ...) could be used, making sure that the transformations are spread evenly across the article.

General Text Transformations

The simplest, removal of capitalisation information, has already been tried: SO reports one competitor's experiment where the evaluation corpus was analysed in upper case, with only a 10 point drop in scores. The next simplest is removal of strong cues, such as corporate designators and person titles.

Company names are sometimes identified by use of novel words or abbreviations, and person names by common forenames and surnames. We can use common nouns or common surnames for companies, and remove common forenames or replace common surnames with unusual ones for people.

Verbs and nouns can be replaced by related words, such as synonyms or hypernyms. This may be implemented using a database like WordNet [Miller, 1990]. The sheer volume of combinations possible will mean that this transformation has to be applied selectively.

Key-based Transformations: Coreference

Coreference chains link mentions of the same concept, supposedly, so we can exchange markups in certain cases. We may also be able to replace complex phrases with their MIN strings; this may actually simplify the text and result in better scores, but the results could still be interesting.

However, an examination of the Coref keys shows that most coreferences are between identical strings and anaphora. For example, 'Smith', 'he', 'his', 'Smith', 'he', ... Exchange of identical words has no effect, and there are many restrictions for exchanging names and pronouns, such as avoiding cataphora. This means that there will be very few useful exchanges of markup.

We conclude that transformation by swapping coreference markups is of limited use.

Key-based Transformations: Template Information

These will focus on the text fragments in the keys: the other information is hard to convert to text changes. Our earlier suggestion of recording where a template string fill comes from may help the application of transformation, ensuring that the intended string in the article is being changed.

Coreference-style information is available from multiple fills in some TE slots, such as the aliases and descriptors. Exchanges among the alternatives should be possible, as these are fairly self-contained. As for coreference, it appears that there will be very few interesting exchanges, so we conclude that transformations based on TE keys will not be useful.

There seems to be little usable information in the ST keys. The only text in the keys are in the POST slots. Substitution among these may change the meaning of the text, especially if there are multiple references to a job in a paragraph and we change only the occurrence cited in the template. So, transformations based on ST keys look unpromising.

6.5 Conclusions

We discuss how well MUC-6 met its goals, consider the weaknesses we found, and then summarise the suggestions made on how to improve MUC-style events, and on how to analyse results. It must be remembered that NL task specification, and automatic scoring are difficult problems, so we cannot make many criticisms in these areas. We must publicise these inherent difficulties, in the same way that the statistical nature of the results are highlighted when reporting MUC-6 results.

6.5.1 MUC-6 Goals

MUC-6 achieved most of its goals. We will not comment on its achievements (see GSD and SO for this), but we consider the weaknesses with respect to the stated goals. It is beyond our remit to consider the wider goals.

The main point must be on encouraging "Deeper Understanding". Firstly, only a limited version of one of the planned three tasks was run. Secondly, in what sense were competitors *encouraged*? There was no pre-defined standard of adequacy, which systems were expected to reach – so no framework against which a system's performance could be interpreted. In the previous chapters, we have seen that parts of a task can be done with surface techniques and simple heuristics, such as coreferencing similar proper names. The organisers have not analysed the difficulty of a task, either before the evaluation (a theoretical sense), or afterwards by some practical measure – eg by the methods used in section 5.6.

Thus, the claim of requiring deeper understanding is unjustified. We have suggested several ways in which this deeper understanding can be measured, which are summarised later in this section.

6.5.2 Problems with MUC-6

These are the weaknesses we have observed. They are additional to the ones discussed in SO (summarised in section 6.1.2 and in occasional comments afterwards). There is no particular order.

- The "Tokenisation Rules" document is referenced in the IE and NE task specifications, but is not published in the MUC-6 proceedings. It specifies some rules on punctuation which are made redundant by the scorer's policy on post-modifiers.
- The scoring method requires detailed definition of the task. This detail in task definitions is difficult to keep consistent and complete.
- The task definitions contain distinctions which may not be important for real users, such as those based on phrase types. A similar situation is with seemingly arbitrary restrictions on what may appear in an answer.
- SGML may not be the best representation for Coreference results. It is converted to template form for scoring, and usually to some other form for visualisation.

- Is there a need for a multi-purpose template design for answers? We argued that the requirements for this are conflicting (see section 6.2.2); the suggested alternative is discussed below.
- The Scenario Task contains an instance of providing scorable features without increasing the information content of an answer (see section 6.2.2). This circumstance also enables expression of contradictory information.
- The representation of optionality and ambiguity in responses is weak, especially in ST. Systems cannot represent uncertain results, so must lose marks on recall by under-generating, or on precision by over-generating. A related case is where systems cannot successfully integrate information: restrictions on template contents sometimes leaves systems with no way to express successfully analysed information.
- Although the ST and TE tasks share the definitions for ORGANIZATION and PERSON templates, their keys are inconsistent. Fortunately, the difference is small, but it does indicate a methodological weakness: the organisers did not check this consistency when measuring Inter-Annotator agreement.
- There is a special treatment for IN_AND_OUT templates in ST, where the key may contain content-identical instances with different parents. It is needed to supplement normal alignment, which works by content alone. This scheme would be useful for responses, where poor system performance can produce templates whose alignment score is identical.
- Unmapped templates in key and response are arbitrarily paired, to convert over-generation and under-generation to simple errors. This is partly a consequence of the heuristic alignment algorithm, which can make mistakes. It also obscures real problems for systems that do under-generate and over-generate, hence does not help diagnosis of problems.
- Over-generation from irrelevant articles in the ST task is not easy to detect from the score tables. A separate statement of this figure, distinct from the scores on the relevant article, would highlight this performance problem.
- The NE scorer allows markups with incorrect text extent to have a correct type. That is, if the string in the key is not marked exactly, but the response postulates an overlapping string with the expected markup type, credit is given for the type. This seems contrary to the task definition: we expect no credit to be given.
- The Coreference task definition appears to need revision in the case of Time-dependent identity (section 5.4 of the definition): "Two markables should be recorded as coreferential if the text asserts them to be coreferential at ANY TIME". This suggests that occupants of the same job are coreferential, which is clearly impossible.
- Some annotators attempted to represent alternatives in the Coreference MIN strings, which was not sanctioned by the task definition.
- The template design for ST contained some elements which were hard to interpret in the responses, for example, OTHER_ORG must be interpreted in conjunction with NEW_STATUS and ON_THE_JOB. The problems with interpretation are suggested by the number of annotator comments in the keys, which explain the templates. The duplication of information with the OTHER_ORG and REL_OTHER_ORG slots was mentioned above.
- Template alignment was a particular problem for ST, especially when errors in response templates made them confusible, under the current scoring methods, with other erroneous templates. Confusion between identical text strings was common, eg when the article discussed several presidents.

6.5.3 Suggestions for the Implementation of MUCs

These are suggestions concerned with the implementation of MUC and similar evaluations. They are not independent: adoption of some suggestions may make some others redundant.

- Representing strings by using direct references to the original text, to reduce alignment problems in scoring, and to aid linking of IE results to the original text. This kind of referencing is implicit in the SGML tasks, hence easily implemented.
- Reducing the amount of detail in templates, leading to a finer grain of representation of results. Using direct references to strings should reduce the alignment problems experienced on small templates. A finer grain will provide more ways for a system to express incomplete analyses.
- Abandon the idea that one all-purpose template design is required. We suggest a simple, logical representation of underlying information, from which human-usable views may be derived. This representation, since it is not intended for direct human use, can be extended to represent ambiguity and uncertainty. One could base atoms of the representation on specific strings from the text. Scoring can be on the simple representation, if the transformation to human-oriented form is algorithmic.
- Automatic Annotation: providing task keys is expensive. A method is suggested for using the competing systems to provide keys. It is inspired by the pooling method of TREC IR competitions, and by qualitative comparisons of performance on the MUC-6 tasks. At the very least, some systems will agree on the correct answer. Disagreement between systems will suggest cases for manual checking. A scheme is envisaged where the output of such a method assists an annotator to produce keys more efficiently. We believe the suggestion deserves more research.

6.5.4 Suggested Methods for Result Analysis

- Weighting can be used with the score results to highlight certain aspects of a task. Intuitively, some parts will be more valuable than others. Deciding on weights is non-trivial, however; it may depend on the reasons for running an evaluation. One objective way of determining weights is by considering relative performance (see next point).
- Correctness Analysis: the score tables indicate performance on an absolute scale, and one may compare the resulting numbers for different systems. Another possibility is to analyse results by how many systems got part of an answer correct. This provides interesting information on the nature of a task, and on how systems attempted this task. The method was explained, applied, and analysed in section 5.6.
- We observed that some aspects of a task are simple, and can be captured by straightforward rules. Performance on those aspects obscures the harder parts, on which non-trivial NLP is required, so we argued eliminating the easy parts in some way. Several methods are possible: manually annotating features in the key with some indication of their difficulty, implementing the straightforward rules in a simple program and eliminating the program's output from the scorable features, or eliminating on the basis of how many systems got the answers correct (ie, correctness analysis). The latter was argued as being the most principled way of implementing a baseline.
- Text transformations: an alternative to a baseline is to change the text so that the cues that the simple rules use are eliminated. This does not reduce the amount of scorable material: instead, it makes the task harder. It also makes better use of the texts and keys, which are expensive to provide. A set of general rules were presented, and rules based on information in the task keys were considered. The former are easy to apply, and appear useful, but the latter are limited because of the few cases in which transformations can occur.

- The overlap between ST and TE, and the different natures of the tasks, suggests that ST scores could be boosted by the easier TE components. Removing them will give a different view on ST performance. An informal analysis confirms a small difference. Correctness analysis on the filtered responses may yield more interesting information.

Chapter 7

Related Work

7.1 Introduction

This chapter reviews the literature relevant to the problem of evaluating general NL systems, as outlined in chapter two. This literature falls into two rough groups: the *practical* – work that has been done on specific systems; and the *theoretical* - considering what should be done. Mirroring the order of this thesis, the practical is presented before the theoretical, with a view to contrasting the two. It is accepted that most practical work shows a good degree of prior planning; our division is intended to separate the mainly theoretical work (ie, that does not have an extensive empirical basis) from that which has. The chapter ends with conclusions. Discussion of terminology is kept to a minimum, the exceptions being where the definitions affect the related frameworks, or where the author finds them controversial. The review mostly ignores system evaluation results: the adequacy of method is more important.

Overall, there is not much literature relevant to our specific problem. There is comparatively little work on evaluation, given the age and importance of the subject. [King, 1996] suggests several reasons:

- the protection of commercial interests (ie, confidentiality) and the belief that third parties are not highly interested in internal evaluation results.
- The traditional form of peer review in academia. Results are usually confidential. The openness of the DARPA events is rare.
- With no good methodology for evaluation, researchers are reluctant to risk criticism by publishing their particular techniques, or by publishing their results. Several sources report the deleterious effects of one early MT evaluation.
- The apparent expense of evaluation means that practical work is limited.

7.2 Practical work on systems and components

Syntactic and semantic analysis are two of the main stages in LOLITA's analysis, where most of the processing is done. We consider work on evaluating performance for both of these tasks as separate components. Generation does not have much effect on the analysis problem, and Morphology is relatively simpler than parsing and not as problematic, so we do not consider work on them. Evaluation of the higher analysis stages (eg Pragmatics) does not seem to have been studied.

Work on evaluation resources is represented by a discussion of the TSNLP project. We then turn to evaluations of specific systems, in particular, those not competing in MUC or who have published evaluation work independently from MUC. There is not much work on individual

systems. Evaluation on specific tasks is considered next: machine translation and information retrieval. The other main NLP task for which significant evaluation work has been done – information extraction – is covered by a few comments on MUC, which summarises the discussion presented in earlier chapters.

Several parts of "Survey of the State of the Art in Human Language Technology" [Cole *et al.*, 1996] are cited in the section on practical work. It is interesting that the chapter on evaluation does not follow the pattern of the overall survey: that is, not all areas of NL research have significant evaluation work to report.

7.2.1 Parsing

[Black, 1996] considers the state of the art in parsing evaluation[1]. He states: "Until recently, objective evaluation was not practised at all, so that even the author of a parsing system had no real idea how accurate, and hence how useful, the system was". ParsEval [Black *et al.*, 1991] provided a *syntactic* way to evaluate broad-coverage parsers, but it is an "extremely coarse-grained tool". It achieves universality by ignoring information in a specific parse from a particular system, such as phrase type and finer distinctions of bracketing. This universality is a reflection of the "fairly superficial" agreement on details of linguistic description between ParsEval's creators. Thus it is comparing parses "at a high remove from the actual parses being judged, and in terms rather foreign to their own vocabulary of linguistic description".

Black notes a rising acceptance of the importance of a good methodology and the rigour entailed by one, and lists several key points. Use of such a framework should be coupled with "well-founded" system-specific measures, thus discounting the idea of comparing systems under a common basis, or under "compromise-based tools". Developers are much better at defining good measures for their own systems. He observes that learning systems based on a common tree-bank may use a measure of tree-matching as a good basis for comparison. We note that all of these points appear in some form in evaluation literature – ie, there is no radically original point.

Another concept discussed is of the "value added" to a larger system by a parser: deployment in a number of larger systems will provide a better evaluation for the parser than testing on an artificial task. Note that Black does not suggest how the larger systems are to be evaluated, nor indicate how many embeddings would be sufficient for a useful indication of a parser's worth.

To summarise: Black, one of the originators of ParsEval, is now unenthusiastic about it. He sees system-specific evaluation, with a strong methodology, as one way forward. Evaluation of parsers inside larger, real systems is another possibility, but he does not suggest how to evaluate such systems.

7.2.2 Semantic Analysis

Following ParsEval, there are plans to design a semantic version, 'SemEval'. However, there is much more disagreement on details among interested parties than was seen for ParsEval, so the result is likely to suffer *more* from the faults observed with ParsEval. Part of this design was done in the planning stages of MUC-6 with the word sense, predicate-argument, and coreference tasks. As seen in section 6.1.1, only a limited form of coreference survived: definition of the tasks and production of answers was found too difficult – there was little agreement on guidelines, and intuitive ideas that worked on simple examples foundered on more realistic examples. Note that the proposals considered in MUC evaluate analysis *up to and including* semantics, rather than evaluating the semantic function alone. Thus, it depends on some form of parsing &c.

The FraCaS project ("Framework for Computational Semantics") in [FraCaS, 1995] considers the main formalisms for semantics with a view to comparison. It has an 'evaluation' of the CLARE and Verbmobil systems. Evidence for the former is taken from [Alshawi *et al.*, 1992]; the source for the latter is not stated. Information on Verbmobil may be found in [Verbmobil, 1997].

[1] All quotes in this immediate section are from this source.

The evaluation takes the form of brief comments under both linguistically relevant headings (eg Anaphora, Scoping) and computationally relevant headings (eg use of Under-specification). There are no conclusions.

To conclude: there are no strong ideas for evaluating semantic components. The formalisms used are diverse and it is not certain that an acceptable commonality is possible. Furthermore, *defining* the notions for a semantic evaluation has been found hard.

7.2.3 The TSNLP Project

This is a recent, high-profile project in test suite construction. The following comments are based on [Oepen *et al.*, forthcoming] and [Balkan *et al.*, 1995]. So far, we are not aware of the publication of an application of the test suite – though some work is in progress, eg an evaluation of a German HPSG parser.

The main aim was to provide a concrete and freely available implementation of a resource many in the field think is useful. More specifically, the group aimed to construct a large test suite with detailed annotations for three European languages, and with software tools to support its use. Construction of test suites is a non-trivial enterprise, and there was little work to use as a basis, so a methodology for creating test suites was developed.

A first limitation was to syntax, and there is much syntactic detail – more than most current IE systems would need to consider. Furthermore, "Semantic and Pragmatic phenomena are less accessible to the test suite method" [Balkan *et al.*, 1995]. Next, the researchers attempted to use a limited vocabulary and to avoid categorial and semantic ambiguity where possible (not explaining what this meant). Care was needed to ensure that phenomena were isolated to ensure that only one thing at a time was being tested – this is particularly difficult in NL. In particular, "Sentences are also kept as short and simple as possible, by, for example, using declarative sentences in the present tense and avoiding modifiers and adjuncts" [Balkan *et al.*, 1995].

The result is over 4000 test items in English, French and German, covering nine classes of syntactic phenomena and two extra-grammatical classes[2]. The form of the test items promotes parameterisation, which could extend the test set automatically by providing legal and illegal instantiations along certain parameters. Currently, only grammatical feature restrictions can be varied.

The TSNLP group admits some weaknesses in the work. The main problem is assessing coverage against a corpus. Natural text contains many phenomena which often interact significantly and may even cancel each other – such as local ambiguity being resolved in a wider phrase context. Thus, there is a problem of matching test items to corpus material. The group suggest tagging the text and matching at an abstract level. Another notion is of test item *importance*: this is a measure of its frequency. The matching problems apply again. However, frequency does not imply relevance – where an infrequent item could seriously affect interpretation of a text – so manual judgements cannot be eliminated from the process. More research is required [Oepen *et al.*, forthcoming] (p. 23).

Our criticisms involve the restrictions and assumptions the group have used. The test suite appears of limited use to systems like LOLITA, as they deal mainly with syntax, and avoid semantic ambiguity &c, which LOLITA is designed to handle. The restriction to simple texts does not help work on real texts that contain long and complex sentences. These restrictions seem inconsistent with the general titles of the two papers reviewed.

It would be interesting to see how test items can be combined to extend the test set. The size of the test set, which we can assume is not exhaustive in even the phenomena of small sentences, illustrates how many cases a complex NLP system may have to handle. This number will increase greatly when longer, more complex sentences are considered. Such a test set would also be unmanageably large.

[2] Tense, Aspect and Modality, Complementation, Negation, Modification, Sentence types, Agreement, Word Order, Diathesis, and Coordination, plus parentheticals and abbreviations.

7.2.4 SRI Cambridge's CLARE

CLARE has interest for the LOLITA group as the aims of the two systems are similar. Hence, work on evaluating CLARE may be useful for LOLITA. [Alshawi *et al.*, 1992] contains two kinds of evaluation: a quantitative investigation on processing ability, and a few summaries on applications of CLARE/CLE. The latter are reports of work done or planned. Three areas of performance are investigated quantitatively: basic coverage, dependence on text type, and effects of sentence length.

Test data and criteria

Parts of the LOB (Lancaster-Oslo-Bergen) corpus were used, as "the closest available approximation to a general or 'unbiased' set of sentences" (p. 226). Sentences with uncommon uses of punctuation were excluded, as were sentences exceeding twenty words in length, to form a 'legible' subset. "Longer sentences stand little chance of being successfully processed by the system" (p. 226). Under half of the corpus remained, with a much reduced average sentence length (mean = 10.9).

A further restriction was made in some of the experiments (the "vocabulary-limited" cases), to those sentences needing only the core CLARE lexicon: 78% of words in the LOB corpus were in coverage, but only 9% of legible sentences contained no out-of-coverage words. Obviously, this further reduced the average sentence length. The vocabulary-unlimited cases were run with CLARE's lexicon extended by a less specific lexicon for non-core words.

The corpus filtering destroyed context, hence discourse analysis such as anaphor resolution could not be expected to work well. Parsing failures would also lose valuable context information. Hence, only results up to the end of semantics were considered. The result of semantics is a 'qlf' – "quasi-logical form", essentially a first-order logic term containing unresolved references, ie lacking a context.

Two main criteria were used: production of a qlf, and producing a good qlf as the most favoured analysis. A good qlf is defined as being correct in some reasonable context. This was done manually on a subset of results, and *estimated* for the complete results. The estimate was defended by noting that the conditions are not realistic, so the results should not be taken as wholly indicative of system performance. The production of any parse was used in a minor way in later experiments.

Basic Coverage Changes

Graphs of percentage for which a qlf is produced were plotted over sentence-length class for the vocabulary unlimited and vocabulary limited cases, and for four successive versions of CLARE. The "good qlf" performance was tested by manually assessing one hundred qlfs from each of the eight conditions, and multiplying this percentage by the percentages from the first experiment. We need not comment on the exact results, but the much poorer performance on the vocabulary limited case for both parts of this experiment is very interesting, as is the discussion it elicits.

The poorer initial production rate is explained with the observation that "in any natural language, core vocabulary items tend to be those with the most complex behaviour" (p. 234). This is reasonable, given the evidence from lexicographers of a small subset of words occupying a disproportionate amount of a dictionary. "Thus the restricted vocabulary sentences contain many words with a large range of syntactic and semantic behaviours, only the most frequently occurring of which [behaviours] are represented in the core lexicon." Performance should be much better when texts contain domain-specific words and concepts, and when CLARE is tailored to the domain.

This suggests that the nature of language in a specific domain *constrains* the problems of NL analysis, certainly at the levels of syntactic and semantic analysis. That is, analysis is easier when a sentence contains domain-specific phrases and meanings that the system is explicitly configured to recognise. Alternatively, a good analysis of domain-independent language is a lot harder than

of domain dependent language.

A related point: the disproportion in the percentage of good qlfs for the unlimited case may be due to the external lexicon containing less detail than the core lexicon, so the qlfs produced are less detailed and hence more vague, which in turn can be more acceptable in an arbitrary context.

Text Type and Sentence Length Dependence

Five corpora (LOB, ATIS, plus three others) were filtered for legibility and length (10 words or under), and a subset prepared for a limited vocabulary case. The median sentence lengths were 4-5 for limited, 6 for unlimited, which greatly reduces the usefulness of the results. No strong difference in performance on corpus was seen for either case, for produced qlfs or for good qlfs, except for the ATIS sentences where the sample for estimating goodness of qlf was too small since few qlfs were produced. So, this experiment cannot suggest dependence on text type for more realistic texts.

To investigate effect of sentence length, four hundred sentences of *each* sentence length of one to twenty were chosen randomly from the legible LOB subset. The proportion parsed and proportion producing a qlf were plotted against length. An unsurprising decrease is seen with length, and it is almost linear. The proportion which parsed was also plotted: this graph is slightly below the qlf graph. This indicates that producing a semantic result from a parse succeeds on most occasions.

7.2.5 University of Rochester's TRAINS-95

TRAINS is designed to be a conversationally proficient planning assistant, currently working in the train route planning domain. In particular, it is described as a dialogue system. [Sikorski and Allen, 1996] considers how such a system should be evaluated. They reject "technology-based evaluations" in favour of the pragmatic view that "the ultimate test of a dialogue system is whether it helps the user in performance of some task." They note that this shifts the problem from a spoken language system evaluation to an evaluation in human factors and human-computer interfaces. The time for a user to complete a given task and the quality of the solution are their main measures, and they formulate hypotheses relating variables in the system to effects on these measures. These are put in the context of some evaluation goals.

Experiments are then designed to test the hypotheses, and to provide quantitative information which can be used to assess future versions of the system. The experiments are described clearly, including the materials used, such as instructions to human subjects.

To summarise: this is a well-presented evaluation experiment, with clear hypotheses. It takes the pragmatic view that the best way to evaluate their system is through performance on its intended task. Time for a human to perform a task using the system is a key measure.

7.2.6 Machine Translation work

Essentially, there is no real agreement on evaluation in the MT field. Despite a long history (around forty years – it is one of the tasks which motivated NLP), fully automatic MT is not yet possible and focus is moving to human-aided systems or systems which aim to support a human translator (see articles in [Zaenen, 1996]; [Kay, 1996] is quite pessimistic). Consequently, evaluation on the more automatic systems is often done on a 'raw' translation which requires some human post-editing. MT evaluation is almost as old as MT, but there is little agreement on the criteria to use on such raw translations. These criteria are mostly subjective qualities, such as clarity, appropriacy of style (see, for example, [Galliers and Sparck Jones, 1993] section 2.1.1, or [Hutchins, 1996]). Note that the nature of translation means that many answers are equally acceptable for a given text.

This lack of agreement may become a serious problem: "With the rapid growth in sales of

MT software and the increasing availability of MT services over networks there is an urgent need for MT researchers, developers and vendors to agree and implement objective, reliable and publicly acceptable benchmarks, standards and evaluation metrics" [Hutchins, 1996]. Thus, as the technology becomes more widespread, lack of standards can damage the emerging market. Note that Hutchins expects several classes of interest to be involved, although he doesn't specifically mention potential customers or users.

We have a little experience with translation. Simple translation from Italian to English has been demonstrated using LOLITA. An Italian grammar is used to analyse text to a meaning representation (as for analysis of English), and then the system NL Generator produces English from the meaning. It has not been evaluated in any sense, but dedicated systems are probably much better: weaknesses in the analysis and generation phases lead to many errors in the final result. Obviously, MT is a demanding task for LOLITA as it requires high performance throughout the system.

But, translation is a highly specialised task with its own inherent techniques. Few translation systems seem to be built using LOLITA-style architectures and the task does not exclusively require a LOLITA-style implementation. However, our understanding of translation systems is that most could not perform tasks like Coreference. It is possible that LOLITA-style solutions could help solve some of the outstanding problems in MT. In conclusion, the possible degree of specialisation and the state of evaluation in MT makes it inappropriate to import MT evaluation techniques directly for our problem.

7.2.7 Information Retrieval

We shall omit the details of IR evaluation, but note some interesting contrasts with IE work. Details of IR evaluation may be found in the reports of the TREC competitions [Harman, 1996] and [Galliers and Sparck Jones, 1993], among others.

Our comments concern the general state of IR evaluation, and are taken from [Sundheim, 1996]. IR evaluation is now mature, especially with the TREC events (DARPA sponsored, as is MUC) which have refined techniques and metrics to the extent where reliable progress from TREC to TREC may be gauged. IR research provided the general notions of recall and precision to IE, although the MUC definition differs slightly from the IR definition. However, the way they are presented is significantly different: recall-precision curves. These are derived from the basic output of ranking candidate articles by a relevance index. Sundheim argues reasonably that IE systems could not produce such curves, as they do not provide relevance rankings of results.

Also interesting is the difference in size of test corpora: IR tested on thousands of documents, compared to the hundred or so used in MUC work. Obviously, manual production of answers is infeasible for so many texts. TREC-3 uses a 'pooling method' to produce a list of relevant articles, which is estimated around 80% complete. We discuss the possibility of using this method for IE, and NLP in general, in section 6.4.3.

7.2.8 Information Extraction Work: The MUC Series

Since previous chapters have discussed MUC-6 in some detail (especially chapter 6), we just summarise the key points of the MUC events. A task is specified by a panel of developers and representatives from DARPA, and an open invitation is made for systems to attempt this task, with no constraint on method. Correctness is defined by automated (syntactic) match to a human-produced answer. The human answers are produced by people experienced with the task specification, and some consistency checks are performed. This "ground truth" is not "perfect truth" for reasons such as human factors, incomplete task explanations, and the inherent vagueness and ambiguity of text [Sundheim, 1996]. The official result is in terms of statistical significance groupings [Chinchor, 1993]: these groups are systems which are not distinguishable on the *sample* of texts used in the evaluation, subject to the level of confidence used in the statistics. The results are apparently linked to funding in the U.S. ([Cunningham *et al.*, 1995] p. 6). The results for non-U.S. systems may still have an effect on their reputation.

There may be some goals attached to the design and mixture of the tasks, eg MUC-6 included a form of semantic test (coreference) and a marketable component test (named entity) in addition to two more traditional template-filling tasks. There is a general wish to isolate system strengths and weaknesses. This has resulted in more evaluation options for participants [Sundheim, 1996].

Obviously, there are several deficiencies to this form of evaluation, as discussed in chapter 6. But, they do show "how much care, and how much effort, is involved in serious evaluation" [Galliers and Sparck Jones, 1993] (p. 112).

7.3 Theoretical Work

7.3.1 Galliers and Sparck Jones' Framework

[Galliers and Sparck Jones, 1993] is a "detailed analysis and review of NLP evaluation, in principle and in practice". Though it is intended as a general work, we shall concentrate on the aspects relating to systems like LOLITA. GSJ's classification of such systems is discussed first, before we briefly review the three chapters of the report. Then conclusions are drawn. GSJ was stimulated by the CLE and CLARE projects [Alshawi *et al.*, 1992]. An evaluation of CLARE was discussed in section 7.2.4; that work makes no use of GSJ.

Generic and General Purpose Systems

How to evaluate NLP systems like CLE and CLARE is a central concern of the report. The term *generic system* is introduced for such entities. A *generic system* is designed to perform a certain task, or more broadly a task type, in different domains: it can be tailored (by adding domain specific resources) to different applications.

This is contrasted against the notion of *general purpose* systems, which are intended to be directly usable without further tailoring for more than one application. GSJ state: "general purpose systems do not exist even for any one NLP task, let alone a range of tasks", though they later say: "within certain limits, or on certain assumptions about the scope of language processing, generic NLP systems are essentially general purpose".

Observe that these definitions are not exact: GSJ themselves blur the distinction with this last quote and admit that the distinction is crude. They do not provide criteria for when a system of certain *design* can be accepted in either of their classes. For example, one can produce a system and claim it is generic, but what criteria are there for accepting it as generic? There is a big distinction between successful generic systems and ineffectual ones. Furthermore, GSJ do not define tasks, task types, domains, in any detail. We note a general difficulty in pinning down such notions in NL work. This can be because the terms were originally introduced informally as a convenience, or because we lack methods to base definitions on complex natural language. For example, the term 'domain': it is used in the field to distinguish between a system working on a specific task, and one that is claimed to be general, or to explain the difference between two specific systems. We do not have a good idea how to compare domains, or of how to analyse similarities between domains, eg when asking whether one system can perform well on a 'related' domain.

The next question is of where LOLITA fits in. Clearly it is not generic, and not general purpose either. [Smith, 1996] introduces the term *general purpose base system* for a system which is not restricted to a task type, but is not a general purpose system. Nominally, we shall place LOLITA in this class, although following GSJ, under certain assumptions LOLITA can also be described as general purpose.

We end by asking, can a general purpose system ever be produced? If we accept the intuitive definition of a system which can be used without modification, we can envisage that such a system may have to contain and competently use most of the linguistic knowledge in the world. Hence, is the notion of a general purpose system *useful*?

In the following discussion on GSJ, when we make comments about generic systems, they will

implicitly apply to general purpose, general purpose base, and LOLITA-like systems – that is, generic systems and above. This is a conciseness measure.

"The Framework: scope and concepts"

Part of GSJ is based on its authors' experience with IR work. IR test and evaluation methodology "has been painfully developed over the last thirty years" (p. 22). "The test methodology established in the IR field has resulted in reasonable test practices (even if they are less than ideal and not always adopted) in relation to the needs for variation and comparison on variables and parameters, and for adequate sampling" (p. 26). So, IR evaluation work is not perfect, but it is mature and has a good empirical basis.

Terminology from IR work is stated, and then extended to cover GSJ's view of NL systems. They admit that generic systems are "in an important sense more substantial than either the IR technologies or expert system shells" (p. 44). This first chapter is quite wide-ranging, and hence hard to criticise except in detail. We shall concentrate to the main points that affect us.

- GSJ introduce the concepts of 'l-system' and 'n-system' as the language and non-language sub-parts of an application. These are contrasted against the more familiar "back end" and "front end", which GSJ see as trivialising the relationship: "it should not be supposed that a system's use of language can always be decoupled from the rest of what it does" (p. 13). This point is supported by an observation from the Edinburgh Workshop on the Strategic Role of Evaluation, 1992 (p. 110).

  This argues against a key assumption in this thesis, which we defend by observing that we have built a 'core' system with no particular application in mind. This is further discussed in chapter 8.

- They distinguish between extrinsic and intrinsic evaluation. The former covers the system in its widest context, namely that of its intended use, complete with real users, real texts &c. Intrinsic evaluation covers a more internal evaluation, such as by correctness of internal results.

- GSJ claim that only an extrinsic evaluation has real force. This is the view in IR. Note that IR is a specific application as opposed to a general framework such as IE. They use the term 'setup' for the system and its working environment. Thus, evaluations should be of setups.

- They reject the view that generic systems can be evaluated well, or evaluated *thoroughly* by intrinsic methods. The result would be limited, and not indicate likely success in a possible application. Furthermore, success in one application does not strongly suggest success in another application. They admit the possibility of "systematic, if limited, evaluation".

- A further observation in GSJ is on modularity in generic systems. Most tend to make strong assumptions about the relationships between components. Concentrating on the performance of single components allows "the danger [...] that all the more challenging processing is actually pushed elsewhere, eg into the pragmatic or inferential component dependent on task and domain knowledge" (p. 43). They note that acquisition of application-specific knowledge in a suitable form is a crucial step in producing an application. Thus, it is better to evaluate a whole system as we cannot be sure how much each part *really* contributes to the final result.

- There is a tension in their comments on generic systems (pp. 42 and 134). People will not be interested in "generic shells" unless they are shown to be useful, and in particular, likely to be useful for an application of great interest to them. That is, people rarely want generic shells per se, although large clients may like the idea of such apparently customisable software. But developers (in GSJ's view) cannot show usefulness unless a specific application is developed. Construction of a convincing demonstration will not be a cheap or quick exercise, and the success here will not really be transferable to other applications.

- GSJ say: "The main problem in evaluation is finding measures, ie concepts which are both instantiations of generic notions and are measures" (p. 22). Though this seems circular, it does agree with our view that finding specific measures for evaluation is a big problem. GSJ also note that providing good methods to obtain the value of some measure is non-trivial.

"NLP Evaluation so far"

The second section is a detailed summary of evaluation work up to 1993. Details of previous and current evaluation methods, competitions and workshops are discussed, particularly in the areas of machine translation, message understanding, speech recognition, and database query. Their review is thorough, and contains much interesting information. Its conclusions, relevant to generic systems are:

- "There seems to be no escape from the fact that the precise capabilities required of [the] working system will have to be laid down, and then the systems evaluated against these, just as in the MT and database cases" (p. 90).
- NL Generation evaluation faces a problem "of designing tasks so that better generators perform the task better", if a task-based evaluation is desired (p. 99). The point about correctly designing tasks transfers to systems in general, if we are trying to find the 'best' system.
- GSJ comment that many of the DARPA evaluation participants see these as evaluations both of their system in a specific task, and of their system from a generic point of view. That is, participants would like their systems to be seen as suitable for other IE tasks. Indeed, several participants have used variants of one basic system in successive MUCs.
- GSJ discuss the report of Engelien and McBryde (1991) (p. 134) on market systems. Whilst not covering generic systems explicitly, it does show the concerns of commercial users. "It is hard to demonstrate generic systems to potential customers since they require an application before it exists (and BBN are quoted as saying that large clients are primarily interested in generic potential)." This is seen in the MUC series, where an interest in core functionality is reflected in the tasks. Note that large clients would like to obtain a single system which can be easily tailored to the variety of applications they are interested in, rather than a set of different, and probably incompatible, single applications.

  Customisation is seen as a problem: the time and cost estimation is difficult and rarely accurate, it is generally expensive, and too complex to be left to users. Companies are also interested in *general* setup issues, such as running costs, productivity of staff, training, personnel requirements, hardware, &c.
- GSJ see "specific dangers in some current trends" (p. 136). These include the bottom-up attempts to define and determine performance for the invented tasks in DARPA work, and generally, "a failure to consider adequately what and who evaluation is for, and hence to take the implications of evaluation into account in test and evaluation design" (p. 138).
- Their conclusions on actual evaluations are (p. 137):

  A1 Evaluation is strongly task oriented, either explicitly or implicitly.

  A2 Evaluation is focussed on systems without sufficient regard for their environment.

  A3 Evaluation is not pushed hard enough for factor decomposition. (ie, to attribute performance contribution to components).

  A4 Generic evaluation is often inadequately delimited (ie, generic entities are evaluated only in a specific application, and the claim of genericity is not tested. MUC is an example.)
- Observing that exploration is the "order of the day", their conclusions on methodology discussions are (p. 138):

M1 The methodology is primarily task stimulated and oriented.

M2 The methodology is more concerned with systems (or subsystems &c) than environments.

M3 The methodology is not decompositional enough.

M4 The methodology fails to define what is meant by a generic system (or subsystem, component).

The last point is interesting: GSJ themselves do not really define the term well. Indeed, they modify it at points, eg saying that generic systems can be general purpose under certain conditions.

- From their survey, they note the following issues (p. 139):

  I1 What is evaluation for?

  I2 Is evaluation comparative or predictive?

  I3 Can evaluation criteria, measures and methods be generalised?

  I4 How do fixed exemplars and artificial constraints help?

  I5 Should NLP evaluation be linguistically or computationally oriented?

  The first four are questions to be asked of a particular evaluation, whereas I5 is general.

We cannot disagree with the conclusions of their review.

"Strategies for evaluation"

GSJ make two general recommendations. They admit that the recommendations are modest and informal, but defend them as a sound foundation. GSJ observe that the variety possible in the kind of evaluation they are considering means that specific scenarios are not possible, despite some of the evaluation literature suggesting that it might be both feasible and *desirable*. Evaluations have to be carefully designed for each case. GSJ believe that there can be no "magic bullets" for evaluation: "not because we do not know how to make them, but because they would not do the job"[3]. Thus, the recommendations are guiding principles (p. 140):

- "Unpack the evaluation by working systematically through the series of relevant questions, pushing into the necessary detail and decomposing the evaluation subject in corresponding detail for proper answers." These questions, the "Evaluation Remit" and "Evaluation Design" are given as a form to complete for each evaluation[4].

- "Envisage the evaluation from the start as a programme of coherently-related component evaluations based on a performance factor grid."

We view them as conservative consequences of GSJ's review, and reasonable within their framework of evaluation in general. The rest of their third chapter shows an application of their recommendations to a hypothetical application. The conclusion of the report is a 'slogan': "in evaluation, it is always essential to look at the environment factors". That is, "while NLP evaluation as such is fine, it is of limited value: what matters is the setup".

[3] Though, by definition, magic bullets *would* the job. We just don't know how to make them in this case.

[4] This scheme is extended by [Smith, 1996] (p. 218) to include an *Evaluation Review*, which contains the Results, Summary of Methods used, and the Evaluation conclusions. This prescribes the form of the 'output' of the Evaluation. The extended scheme is a more complete record of a particular evaluation.

Summary and Conclusions.

There seem to be several main points, which we shall now discuss.

- **Evaluating a setup**: This conclusion is quite pragmatic. A system is shown to be of value by showing that it is of value in the intended context. Lesser forms of evaluation, such as by internal details of component, do not have this force. Theoretical aims are irrelevant if the major purpose of the system is to do a certain job within specified limits. We cannot argue with this.

- **Their recommendations**: The questions GSJ suggest refer to important issues, the consideration of which is a good idea for each particular evaluation. [Smith, 1996] reports that the framework was useful in an NL Generator evaluation, where there is little established methodology, and is a promising framework for future such work (p. 231).

  GSJ note that filling in the remit and design is more difficult for an artificial setup: there are no natural answers, and the process of completing the remit and design can be misused, shaping the artificial setup by providing answers which look good. This is an argument for realism in task (or setup) identification.

- **Generic systems**: GSJ do not really contribute here to the practicalities of the subject: their view seems to be that generic systems can only be evaluated when customised to a particular setup, and the results are limited in validity to that setup. Generic systems are not considered in their conclusions. GSJ do mention the tension between being general, but needing specific evaluations. They allow the notion of an intuitive worth of a generic system.

  The lack of solution, plus the force with which they argue for evaluation within setups, could be seen as an argument against generic systems.

As to the worth of the work: it is quite abstract, so needs further work before use. The recommendations are sufficiently general as to be widely applicable, and will be hard to argue against. GSJ offers little guidance on the particular problems we face, and asks more questions than it answers. Of course, we need to consider answers to some of these questions *ourselves*. There is little help on our specific problem, of generic systems and above. This also applies to one of the motivations of the report: we are not aware of any application of GSJ to CLARE.

7.3.2 EAGLES Evaluation Working Group (EWG)

The comments in this section are drawn from the EAGLES Working Group on Evaluation's interim report [EAGLES, 1994], their final report [EAGLES, 1995][5], and King's article[6] on reducing costs in evaluation through resource sharing [King, 1996]. Many of the criticisms made of GSJ apply to the EAGLES work, so we do not review the work in detail. We do review the individuating factors, however.

EAGLES (Expert Advisory Group on Language Engineering Standards, [EAGLES, 1997]) is an EC initiative which aims to accelerate the provision of standards for evaluation and other areas related to LE. Well-known companies, research centres, universities and professional bodies across the European Union are collaborating with the aim of producing a set of guidelines in the areas of interest. Participation is voluntary, but resources are limited. Reuse of material is a key point. The basic idea behind EAGLES work is for the group to act as a catalyst in order to pool concrete results coming from current major European projects. This is believed to spread costs of some work across several projects.

[5] The final report was only released to the public late in 1996.
[6] King chairs the Evaluation Working Group.

Introduction

The EWG reports are the result of some thirty months of voluntary effort by around twenty people from academia and industry. It contains a general framework for user-centred evaluation of "adequacy", and detailed investigations into the evaluation of writers' aids and translators' aids. The EWG admit the narrow focus. The interests of the contributors appear oriented towards the latter applications, so we conclude that there is a bias which is incongruous with the generality of the work's title.

Their introduction contains a brief and selective review of "Recent history". It discusses evaluations from 1966 and 1973, makes only a brief mention of GSJ, and on comparative evaluation discusses MUC-3 from 1991 – though there is a discussion of more recent DARPA MT evaluations. Does this literature review contribute to the design of the general framework? It seems not.

The EWG state that the reports are *not* polished, final work – they have not exhausted the subject. However, certain elements are unlikely to change, and we examine these. There are four main areas: use of ISO 9126, the Parametrised Test Bed (PTB), the Consumer Report Paradigm, and Software Engineering in NL applications.

ISO 9126 and its extensions

The ISO 9126 for quality characteristics to be used in the evaluation of software is used as a basis. These are points that can be thought important to identify for a software system. EWG do not justify its use, other than NL systems being a form of software. Software Requirements are seen as the basic input to the ISO 9126 scheme. The EWG admit that ISO 9126 "may need expansion to deal with the special case of NLP systems", but do not make any clearly identifiable suggestions, nor outline why the case is special, other than to note the importance of Requirements Analysis to NLP systems and to suggest that Customisability be made a major characteristic for generic systems. The ISO 9126 characteristics are (quoted from [EAGLES, 1995]):

Functionality a set of attributes that bear on the existence of a set of functions and their specified properties. The functions are those that satisfy stated or implied need.

Reliability a set of attributes that bear on the capability of software to maintain its level of performance under stated conditions for a stated period of time.

Usability a set of attributes that bear on the effort needed for use, and on individual assessment of use, by a stated or implied set of users.

Efficiency a set of attributes that bear on the relationship between the level of performance of the software and the amount of resources used, under stated conditions.

Maintainability a set of attributes that bear on the effort needed to make specific modifications.

Portability a set of attributes that bear on the ability of the software to be transferred from one environment to another.

We note that it is a 'truth' of Software Engineering that just thinking about such categories for a software system can have benefit, even if the answers may not be that reliable or useful: the exercise alone will help to improve the resulting system. Such standards may also be useful as a starting point when nothing better is available. However: there is a danger of confusing preliminary investigations expressed in a formal way, with final results. That is, the provisional nature of the results should be recognised.

Relevant attributes are discovered by analysing requirements documents. A metric must be chosen to obtain a value for an attribute. "Defining relevant attributes is of no use if there is no way to measure the system's value with respect to those attributes." A related concept of 'measure' seems to be the unit of an attribute (eg percentage correctness in parsing). The ISO 9126 Standard quite deliberately leaves aside any discussion of how metrics are to be created or validated.

EWG state: "Measures must be valid and reliable. They must measure what they are really supposed to measure, and they must measure it consistently." This seems circular. The report does not provide much information on how to validate measures. EWG admit that questions of validity and reliability can be hard, particularly when concerned with the complexity of language, where human intervention is sometimes needed to obtain a measurement ([EAGLES, 1995], p. 15).

An example of an unreliable measure is given in [King, 1996] as a task latency where, "through carelessness, some users are asked to read a screen in bright sunlight and others in a darkened room". Another is from [EAGLES, 1995] (p. 30), on which language a system deals with: they note that Greek spelling checkers being evaluated by someone needing an Italian one is nonsense. We assume that most evaluators can avoid such pitfalls of experimental design! The reports contain other such examples.

The Parametrised Test Bed

New to the final report is the notion of a "Parametrised Test Bed" (PTB), which seems a consequence of a related project on authoring aids. A system is "plugged in" and a series of tests conducted by running the metrics implemented in the PTB, possibly with human help to obtain certain measurements. The implementation of non-automatic metrics will guide a human operator through the procedure defined for the metric.

A PTB will therefore represent a standard way of executing evaluations on a single system, which may make comparisons more feasible if standard conditions are used in the data collection. Parts of the reports look like a design for a PTB, outlining the kinds of test and test result (eg results can various types: binary, scalar, multiple choice, ...) – but not supplying details of particular metrics.

The Consumer Report Paradigm

The PTB will *collect* information, but not *interpret* it. Rather than attempting to define some way of combining this information and summarising it, EAGLES suggest a "Consumer Report Paradigm". This means, information relevant and useful to a class of users is presented in a table, as used in consumer reports comparing cars or washing machines. This is to support users in making an informed judgement on the suitability of system &c for their intended use. EAGLES see their work as identifying what the headings of such a table should be.

[King, 1996] admits that consumer reports are intended for stable products: "Transferring the paradigm to the more sophisticated products of the language industry can require a great deal of work, and sometimes a considerable degree of ingenuity". Note the EWG's focus on on-market or near-market products.

They provide no methodology to help users decide overall, such as ranking criteria by importance. This may be particularly important if a system is complex, or many attributes are found relevant. Customers may then be confused by large amounts of information. The problem is worse if attempting to compare systems on the basis of such reports. This lack of guidance is also present in ISO 9126 work – it does not suggest how to make good decisions on the information.

Software Engineering in/for NL applications

EWG make the first explicit suggestion of applying Software Engineering (SE) ideas to NL systems, though it is implied by adoption of ISO 9126 in a sense. They note that requirements analysis is hard for NL systems. Their contribution is more a discussion of current SE work than a detailed examination of the consequences for NL. For example, Appendix B in [EAGLES, 1995] is a survey of testing methods, but it is oriented towards traditional notions of software with very little reference to the typical construction of NL systems. The examples are tailored to the applications investigated in the report. The appendix's conclusions invite others to *test* the guidelines presented within that appendix.

This appendix contains a notable comment on the economics of evaluation (p. 94): "The evaluation budget is naturally the most decisive factor, when it comes to selecting evaluators, subjects, test types and instruments as well as when determining the time that can be invested into evaluation. [...] It is important to note here that, in the case of a limited evaluation budget, it is advisable to reduce the number of metrics that will be tested and to select less expensive instruments, rather than to reduce the number and qualification of test personnel. While a limited number of metrics only reduces the scope of the evaluation exercise, savings in the area of test personnel lead to less reliable test results."

The author disagrees. This seems to say that evaluation budgets may be set arbitrarily and then evaluators have to do the best they can. This does make evaluation seem like an afterthought – which is a common complaint with conventional Software Engineering: that testing is squeezed into the final stage of a project, and done inadequately. Our view is that what evaluation *requires* should be the main factor: there is no point in pretending to do evaluation work if the results are not valued by the people commissioning the work. However, we do not have a good notion of value, or of what is necessary. This is considered in chapter 8. Furthermore, if standardisers do their work well, then evaluation can be reduced to a process which does not require great expertise (and hence expensive personnel) to apply.

Conclusions on the EAGLES EWG Work

- **Provenance and Methodology**

  The report is a voluntary effort; most contributors appear to specialise in the applications considered. There are no contributors that we know to be being actively involved in constructing a large NLP system.

  The aim of the group is standardisation. Two points arise: has enough work been done to be able to standardise? If not, the result is likely to be artificial and of limited use. The EWG admit their narrow focus in actual work – despite the very general title of the report, and admit the lack of previous work to build upon.

  Secondly, it is in the interest of the EWG to produce a framework of some kind. Can we be sure that the framework is entailed by their research? If not, it remains one possible proposal among many which, without more research, cannot be called a 'standard'.

  Our view is that insufficient work has been done, and that the EWG framework is currently only a proposal. Furthermore, the range of contributors is not wide enough to represent the community in general.

- **Ideas Acceptable?**

  The work covers two aspects of evaluation: evaluating existing systems, as in documenting their properties, and evaluating *new* systems. The latter is pursued by a consideration of the Requirements process in NL work.

  A lot of what EWG say is uncontroversial: a mix of good scientific practice (eg clear experimental design, clarity of intention – ie, hypotheses stated) and software engineering, although it is not that specific to NL work. Unfortunately, in the NLP field, such basic information often does need stating: the mix of researchers in the field means that criteria for evaluation and for adequacy of work vary. There is little consideration about the logistics and costs of evaluation, apart from the unsatisfactory discussion of budget balancing mentioned above. The investigations with SE are not as specific as we would like to see.

  Like GSJ, the work is quite abstract – it is a framework for a framework for evaluation. Many of the important 'implementation' details are missing. It appears reasonable on simple NL applications which are market feasibilities. EWG suggest that work on the modest applications lays the groundwork for more sophisticated systems. However, generic systems and the like have not been considered in any depth. There is just one paragraph on generic systems, and this appears only in the Final Report, in which problems of *customisation* are acknowledged as being a *major* factor in acceptability, as opposed to a minor factor subsumed by the ISO 9126 Maintainability characteristic. We cannot be sure of successful scale-up in any aspect of their work.

- **Overall view**

  Overall, the work is not an easy read – it is a large work, with no index. In particular, it attempts to say a little on all possibilities, without reaching concrete conclusions. There are no identifiable hypotheses to test, so the parts of the work of interest to us are mainly opinion. The EWG do not outline criteria for the success of their work, and there have been no independent applications of their ideas that we can consult.

  Parts of it read like a design for their Parametrised Test-Bed (PTB), with statements saying a test result can be binary, a real quantity, a comparison result, a selection from a set of options &c. There is not much discussion of what *actual* tests should be, which we believe is a central problem in evaluation.

  The Consumer Report Paradigm may help with collecting useful and relevant information about a system, but it does shift the hard work of *deciding* on to potential users. Can one expect non-experts in the field to fully understand the amount of information, and the relationships or tensions between classes of information? It seems more reasonable for experts to condense this information, if such a step is possible.

  In summary: the work does not suggest an immediate solution for our problem. There seem to be too many problems to solve before use with LOLITA.

7.3.3 CGN

[Crouch *et al.*, 1995], henceforth CGN, is an interim report of a Study Group on Assessment and Evaluation, commissioned by the European Community (EC) in conjunction with its "Language Engineering" (LE) programme[7]. The report is not official EC policy: it discusses possible guidelines for assessment and evaluation in the framework of the LE initiative. The study group is not independent from the EAGLES Evaluation work (see section 7.3.2), though few, if any, of the ideas in the report appear in the EAGLES documents. Note that the LE programme has an emphasis on the users of resulting products rather than the underlying technology (as may be seen in the EAGLES work).

The mandate was for "the preparation of a proposal for setting up infrastructure and guidelines for technology assessment and performance analysis, including a comparative evaluation where possible", with scope over all projects in the LE programme. CGN uses ideas from four trigger papers on various aspects of Assessment and Evaluation, several responses to those papers, and from a questionnaire. The questionnaire was conducted by the report's authors to elicit more response from the Speech and Language community[8] than was gained for the trigger papers. CGN do not claim thoroughness or scientific adequacy for this sampling of opinion. Unfortunately, there was little response[9] – though this lack of response itself is interesting and significant. This canvassing of opinion from various sources, and the reporting of diverse and often conflicting opinions, is rare in the literature.

Essentially, the result of the report is a recommendation for a European MUC-style competitive evaluation, extended to allow evaluation of intermediate 'technological' results for participants who are interested in those intermediate results (eg parse trees), in addition to the user-oriented end products. This is an attempt to satisfy the many requirements set for the study group.

The finer details of CGN's position will now be discussed, followed by a summary of their questionnaire. The section on User-Centered Assessment is omitted as it is similar to the EAGLES work.

[7] This programme [European Commission, 1996] is aimed at the production of real-world systems; specifically, CGN define an "LE system" as "a set of software components constructed to permit a user to carry out some language-related task or function in a specific real-world environment" (p. 7). This is a weaker and less pragmatic definition than the version adopted for this thesis ([Boguraev *et al.*, 1995] and section 1.1.3).

[8] We assume, the European subset.

[9] "Some ten" – maybe including at least one of CGN's authors – see the first "Other Comment" on p. 76 of CGN.

Background and Terminology

> *When there is little agreement on common terminology, as is the case in evaluation, the definitions invoked are a good predictor of a person's final views.* (p. 8)

Most of the terminology mirrors [Galliers and Sparck Jones, 1993] and [EAGLES, 1995], hence is biased towards single-application systems working in a particular domain, as one may expect from their definition of an "LE system". No distinction is made between Assessment and Evaluation, the terms apparently interchangeable, although assessment is said to comprise Verification and Validation[10]. A distinction is made between "user-centred assessment" and "technology assessment", the former concerning characteristics directly relevant to a user (such as ergonomics, external functionality), and the latter those important to the system but not of direct interest to the user (eg parsing ability and metrics thereon). There is no mention of concepts like 'generic systems' or their relatives, hence no discussion of the issues in evaluating them. We consider two interesting points:

- **Decomposition of Tasks**

  One main point is that sub-tasks (eg parsing, semantic interpretation) can be identified in a task, which need not correspond to architectural modules of a particular system. That is, theoretical objects can be identified in a system's overall processing which need not correspond to a module interface, nor be meaningful to a user. They say: "fortunately, there is fairly widespread agreement about what constitute the main LE tasks, despite theoretical variation" (p. 10).

  CGN note that some form of simple transformation from system-specific representations to a common form is required, "perhaps with considerable loss of information", and claim the existence of "possible" conversion methods (again with considerable loss of information) for different syntactic and semantic representations, but do not give evidence. The problems with such methods and with the metrics defined on the results were discussed above, in section 7.2: clearly, those problems will be inherited by the CGN framework.

  Note that this does not require that *all* parties must agree on the sub-tasks. Instead it provides a framework for a subset of interested parties who share common interests; an evaluation among this subset can then be included in the main evaluation.

- **Task attributes: Depth, Accuracy, and Robustness**

  Among the discussion of possible attributes for a task are the following definitions, as characterising the task's performance of the map from input objects to output objects. They are clearly insufficient. Note that they are described as *task* attributes: but different systems will respond variously in these attributes on a fixed task, providing a good argument for making these attributes partly, if not wholly, dependent on *systems* instead.

  - **Depth**: "A task is done shallowly if certain details of the output representation are not captured, and is done deeply if they are."
  - **Accuracy**: "A task is done accurately to the extent that the details of the output object that are represented are or are not ones that are really there.".
  - **Robustness**: "A task is done robustly if it can produce some sort of output for any input, rather than just failing on some inputs."

Comparative Assessment

It appears that the commissioning body would like a comparison evaluation which tried to vary all factors. CGN question whether the results will be meaningful for a comparative evaluation involving systems in the LE programme without restriction. They also question the utility of

[10] Using the traditional Software Engineering definitions, "Are we building the system right?" and "Are we building the right system?" respectively.

purely user-centred comparisons (ones that make no reference to system internals), and claim of the structure of different systems, "one usually finds that they have tasks and components in common" (p. 32). CGN admit the relevant components can have differences in attributes which comparisons will have to take into account, and will also have to "try to relate performance to user attributes". CGN does not indicate how to do either of these.

Thus, they claim an assessment of the technology is natural in comparative evaluation, but admit that not much is known about the user-centred implications of existing technology metrics like ParsEval. More metrics are required. Several iterations of the evaluation would be required to correlate between results, in order to discover and validate the technology measures. Of course, some metrics or other parts of the evaluation design could be found lacking. CGN's justification of this kind of evaluation appears oriented to a user-focussed commissioning body. Several advantages of comparative assessment are suggested and discussed:

- Cross fertilisation: one feature of MUC evaluations, the similarity of task (and hence, of system goals) promotes adoption of successful techniques. In particular, the nature and publicity of competition avoids the complacency which a technically inexperienced user would not detect.
- Maintainability and flexibility – by using domains other than a system's intended domain.
- Identifying the promising technologies. This is useful, but suffers from the dangers of too much cross-fertilisation. Secondly, the complexity of systems means that promising technologies are rarely implemented (or implementable) as isolatable components. Evaluation metrics would be required to reliably detect and diagnose the reason for a system's relative success.

Next, the form of an EC-sponsored comparative evaluation is considered, examining the constraints provided by the nature and scope of funded projects. A DARPA-style design was seen as unworkable given the constraints. The "open competition" format of DARPA evaluations was seen as a good point, and recommended for any EC effort. The focus on black box evaluation on a single task was not praised, as CGN saw lack of technology assessment (ie, of the internal workings) as supporting "task and application-oriented short-cuts to some degree" (p. 44).

An aside: what is the motivation behind this objection to "short-cuts"? Will users care if the system is sound with respect to some theory of linguistics (or whatever)? We believe they will not, and expect that their main concern is with whether the systems works well. Thus, if these short-cuts improve a system's performance, then they should be used.

These constraints are, however, very diverse and do not appear to be easily satisfied, given our experience with MUC-6. We quote them in full here (from p. 36), as they help to explain the 'Braid' model (see below):

- The exercise should be incremental, allowing reuse of materials and experience from earlier iterations.
- It should have low entry and working costs to encourage participation.
- It should consider multilinguality and multimodality.
- The materials should be cheap, especially the evaluation data and answers.
- The materials should be reusable in other S&LP work.
- The evaluation program for computing performance measures should be relatively easy to provide and apply.
- The evaluation structure should allow both technological and user-centred evaluation.
- As far as possible, the comparative evaluation exercise should sit on top of, and make use of, project internal assessment.

The Braid model mirrors the main idea of CGN. A lattice of sub-tasks is defined across a complete task, with theoretical objects being expected at the internal nodes, and user-oriented results on the outside nodes. Metrics are chosen depending on whether the objects are user-centred or not.

A system can undertake a sub-part of the lattice. It is evaluated on the nodes that are meaningful for the system, and on which the developers wish to be evaluated (eg they have reached agreement with other groups as to how to evaluate at a particular node). For example, a system's parsing could be evaluated against other systems that produce parse trees under a ParsEval scheme.

The Braid model alone does not solve the various resource problems, however. CGN discuss the obstacles of task diversity and domain specificity which characterise EC LE projects. They suggest the provision of "layered corpora", where texts are annotated at various levels of information (ie tagging, syntax, semantics), agreed upon by interested parties. The expense of such an effort cannot be easily avoided.

CGN's Questionnaire

This questionnaire (appendix A of CGN) was not intended by its authors to be scientific or exhaustive, but to provoke more response on evaluation than previous trigger papers had. Furthermore, there was a disappointingly small number of replies (ten or so). That the results are considered at all indicates that the authors attached sufficient credibility to the responders to justify presentation of their views; alternatively, we can take the comments as being from established researchers in the S&LP field. The results and (anonymous) comments make very interesting reading. We summarise the main results, and quote some of the significant comments. Note that the questionnaire has several flaws, such as not providing a complete range of answer options, or having bias in the phrasing of some questions. An example is of asking about willingness to participate in one kind of evaluation, without asking how useful a respondent believed it to be. Some respondents omitted sections of the questionnaire.

- There is strong support for both project-internal and comparative assessment. There was concern about the cost of such evaluation.
- At the project internal level, quantitative technology assessment was preferred to user validation, and there was a majority preference for evaluation at levels below the user's concern. There was support for varying project internal data by linguistic features such as domain (despite the specificity of most projects), though some comments saw this step as premature.
- On the subject of project internal assessment, there was a clear preference for evaluators to be project members (as opposed to external evaluators), but no preference for whether evaluation *data collectors* should be internal or external. Opposing comments cited the expense and bureaucracy of using external evaluators.
- As regards comparative evaluation, there was enthusiasm for both separate speech and written language evaluations, and a combined evaluation. Evaluation at fine levels of granularity was supported. Participants were then asked to prioritise user-oriented tasks, subtasks in possible tasks, and possible dimensions for generality. For the first two, opinion on speech was more clear than on written language. Domain was seen as the most important variable for generalisation, with no clear preference on the remainder.
- On the question of how to implement a comparative evaluation, opinion was split between a DARPA-style 'top-down' design where the task is set by a central body, and a 'bottom-up' style where participants suggest and refine a task. Objections were strong here, at extreme doubting the feasibility and sense of the rejected option. Argument suggested that projects were too diverse to make *either* option workable. So, there is *no consensus* about how such an exercise should be implemented.

We now present a selection of anonymous comments from the questionnaire.

- p. 66: "For more theoretical work, [project internal assessment] is more problematic since you might not know what to measure, and it may change as the project progresses."
- p. 68: "Quantified numbers tell the truth, or appear to."
- p. 68: "It is well known that most users do not know what they want or need."
- p. 68: "[...] depending on users' existing preconceptions introduces a huge inertia into technological development."
- p. 69: "Difficult to meaningfully evaluate at [sub-task levels]."
- p. 77: "Administrators will find it convenient to have numbers that can be put in rank order."
- p. 77: "I think there's a great risk of stifling new work that doesn't fit the pattern. An occasional open competition of the DARPA sort sounds like a fine idea, but making it a model for all work does not. There is also too much scope for creating a class of professional evaluators, who will be people that couldn't make it in research. There are already some groups like this around. The first item on their agenda is, naturally enough, to keep themselves in business, which is not the same as promoting good work."
- p. 77: "In a small community like this, people with enough expertise to be of use [as external evaluators] are very likely to be either colleagues or competitors – or a bit of both – of the people being evaluated."
- p. 77: "It's not obvious whether your questions should be answered from a realistic or an idealistic point of view ..." "There's too much of a top-down flavour about things, as if one is saying, OK lets [sic] evaluate (for its own sake), so what's the best way of doing things. A more appropriate view would be to have some desiderata and consider various possible evaluation suggestions or scenarios against these: this seems to me the only sensible way of getting a fix on what LP systems/components are of value in relation to the conditions which make them of value."

From this small response, we tentatively conclude that there is an agreement of the importance of both user-centred and technological evaluation (in CGN terminology), but little consensus on how to implement it. Opinion is not as homogeneous as one may believe from the standard literature. Obviously, this is not good for our practical concerns, but it does confirm that there is still a lot of work to be done in the topic.

Summary

The main recommendation is for a DARPA-style *open* competition on the basis of their Braid model. CGN outline a possible route to establishing such an event, including a discussion of the resource implications: "it should be recognised that such an evaluation cannot come for free" (p. 59). We note that an open field (of competitors) with a commitment to supply funding could entail supply of funding outside the EC.

CGN's recommendation does not appear useful for our problem of evaluating general systems. It is only an outline of a framework, and lacks many of the lower-level details which we find are important, based on our MUC-6 experience, such as concrete ideas about metrics. The scope of the proposed evaluation is so wide that we have concerns about the interpretation of results, and about their utility.

Finally, it is interesting that a *merger* with DARPA is not proposed. This could reduce some of the resource problems, and ameliorate some of the perceived flaws in MUC-6 by closer involvement in the planning stages of future events. The NL community is probably not large enough to support two large competitive evaluations.

7.3.4 Allen's comments in 'NL Understanding'

Allen's book [Allen, 1995], a standard in the field, contains a few comments on evaluation which we have not yet encountered.

- Black Box evaluation in the early stages of work can be highly misleading. "Only when the success rates become high, making a practical application feasible, can much significance be given to overall system performance measures."
- Care should be taken with systems that can rely on the user's intelligence to appear intelligent themselves. For example, the famed case of ELIZA ("a collection of tricks"): its apparent capabilities had more to do with the naivety of users than with NLP. Thus, objectivity is required.
- His conclusion is that "either we have to accept certain theoretical assumptions about the architecture of NL systems and develop specific evaluation measures for different components, or we have to discount overall evaluation results until some reasonably high level of performance is obtained. Only then will cross-system comparisons begin to reflect the potential for long-term success in the field."

7.4 Conclusions

We summarise the main points of the review, and then draw an overall view.

7.4.1 Summary

Component Evaluations

Comparison between the components of different systems involves compromise, and it appears that too much compromise loses information. Compromise-based evaluations exist only for the lower levels of analysis: disagreement on the higher levels inhibits the formation of proposals for them. There is a preference for project-internal, and system-specific, evaluation, coupled with a strong methodology, which should include clear, a priori definitions of result acceptability, objectivity, and publication of results.

TSNLP

The project has constructed three parallel test suites, but we have not seen concrete results on its use. The suites are oriented towards syntactic work; the annotations involve compromise on notations and theory. It is admitted that test suites cannot be easily constructed for semantic and pragmatic analysis. Furthermore, there is no ranking of importance for the test items, so there is no strong link from test success rate to performance in actual use. Furthermore, the relationship of performance on simple, isolated cases to performance on sentences which contain many, possibly conflicting, cases is currently unknown.

System Evaluations

We report on two. CLARE is evaluated along several dimensions on internal structures (parse production, qlf production, qlf acceptability). Acceptability is estimated by experts, as being reasonable in some context. It is not rigourously defined, and some conditions (such as use of an external lexicon) raise doubts about the notion. An interesting result is that sentences not containing domain-specific vocabulary are harder to analyse. The performance of TRAINS-95 is translated to latency time and quality of result, both user-oriented quantities. It is argued that task performance is the ultimate test of a dialogue system; this shifts the problem from system evaluation to an evaluation in human-computer interfaces.

Task-based Evaluations

We briefly considered MT, IR, and IE. MT was seen as too specialised with respect to the demands made on LOLITA to be a useful evaluation task, especially if competing against systems designed specifically for MT. There is also much disagreement in MT evaluation. IR has a developed methodology, some of which has been imported in IE work. However, IR systems are different in nature from NL systems: eg, IR results allow production of a recall-precision curve whereas NL systems usually produce discrete results.

IE tasks are usually scored by syntactic matching to a human-produced, predefined answer. This kind of scoring is not very flexible, so task definitions must be precise about answers. Consequently, machines must be similarly precise. There are difficulties in applying the scoring methods to linked templates rather than flat templates, but problems in representing complex information with flat templates. IE is a reasonable task for LOLITA-like systems, although good results can be obtained by techniques simpler than LOLITA's normal style of analysis.

Competitions

We mainly consider IE competitions. Competitions involve several systems, which are compared on the output of several tasks. The tasks are are usually designed by a committee of developers and customers, and the scores are obtained by a syntactic marking procedure. Statistical measures are employed to indicate the significance of differences in scores. The results are made public. The competitions have a good infrastructure: a central body coordinates the event, provides training data, produces answers and scoring software. The competitions also focus interest, and have prestige.

Evaluation Theory

- **GSJ**: presents an overview of previous evaluation work, and suggests how evaluation of NL systems should be done. Given their analysis of previous work, and a consideration of what they call "generic systems", their suggestion amounts to the providing a clear statement of the aims and standards for a system-specific evaluation. They conclude that no universal method is possible, given the conceivable variety of NL systems. They also suggest that generic systems cannot be evaluated properly on their underlying language-handling ability, but should be evaluated in the context of actual use for a task. GSJ acknowledge the tension between this and the intended generality of such systems.

- **EAGLES**: suggests a framework based on an ISO standard for software quality, and applies it to two basic NL applications. Their interest is more in user-oriented aspects of NL software. Their work is more relevant to near-market products, and can be seen as a listing of facts about a system, with no guidance on assessing the overall worth of a system from this information. They virtually ignore the question of general systems.

- **CGN**: provides recommendations for an EC NL evaluation exercise. They are set unrealistically wide goals, of covering all NL work in the EC, which is clearly too general. They argue for a competitive evaluation which attempts to cover user-oriented and technological (internal) aspects of systems. This is in the form of a 'Braid' model; a system may be evaluated at points in the lattice where its developers agree with other developers on standards for evaluating the internal objects represented by those points. We conclude that the suggestion is weak in the light of experience with component-based comparison (eg ParsEval), and that their suggestion lacks the detail required for successful implementation.

  One very interesting part of the work is a questionnaire on evaluation; more interesting still is the small response to it, and the answers obtained. Despite the flaws in the questionnaire, it appears that there is general favour for evaluation, but opinions are quite varied on what should be done and how to do it. Some respondents expressed concern about current trends in evaluation work.

- **Allen**: warns against premature evaluation for research work, and highlights the "ELIZA effect", where a system makes use of the user's interpretation of output to appear 'intelligent' or useful.

7.4.2 Overall Conclusions

We conclude that the problem of how to evaluate LOLITA-like systems is still open. This class of system is rarely discussed, and the few discussions that exist are insufficient. One reason for the weakness is the difficulty of accurately characterising such systems. This mirrors a general difficulty in quantifying concepts in NL work, for example, the finer points of task design.

Also noted is the lack of agreement in the field. In particular, there are many unproductive arguments between extreme positions. The debate about term-ism in evaluation competitions is one example: the nature of competitions focusses on short-term goals and solutions, at detriment to longer term research, but such long-term research needs serious evaluation. Furthermore, the arguments are often not much more than opinion, and are given in long and complex documents.

There does not appear to be a universal solution, as GSJ observed. Obviously, sensible compromises are required, so we should publicise this, and then look at how they can be arrived at. It is likely that eventual schemes will be based on the *consensus* of all interested parties.

Chapter 8

How Can We Evaluate LOLITA-like Systems?

8.1 Introduction

We concluded in the literature review that little work had been done on the evaluation of systems like LOLITA. There are problems with the evaluation of NL systems for single tasks, as we have seen from the analysis of MUC-6. Therefore, there is scope for investigating the notion of evaluating LOLITA-like systems from basic principles. In doing so, the basis of evaluation in task-specific systems can be re-examined.

The intention in this chapter is to outline one developer's viewpoint of evaluation, examining relevant theoretical and technical issues, based on several years experience with a working system. There are no concrete suggestions on evaluation experiments to perform, but several new aspects of evaluation are brought into the debate. It is the author's contribution to the process of reaching a consensus on evaluation.

8.2 Methodological Comments

The author's view is that evaluation is a hard problem, and that the way forward is to encourage productive discussion between the interested parties in order to reach a consensus.

One possibility for this chapter is to suggest tasks which address the distinctions between LOLITA-like systems and more conventional systems. The analysis of MUC-6 shows how much work is required in good evaluation proposals, and how difficult the work can be. Several issues have been raised during this analysis, from scoring of the output to interpretation of the results; all of these are open problems and will require much work or discussion to resolve.

One may ask, what would be achieved by suggesting new tasks? Many problems observed in MUC will be inherited, so the new tasks will not be immune to criticism. Furthermore, other groups would not be under obligation to attempt those tasks. Basing existing work on a firmer foundation seems a more reasonable goal.

This chapter takes the form of a discussion of several aspects of NL systems and of how one may understand their performance. Some consequences of these views are outlined. The ideas presented have not been encountered in the literature, but the author believes them relevant to discussion, especially as they affect how one views evaluation results.

The author intends that this work will form a set of points which evaluators will need to consider. Future work by others will either extend this framework, or provide suitable counter-arguments to refute it.

What methodological criteria can be set down for such work? It is effectively a set of assump-

tions or observations, with some supporting argumentation and discussion of a few consequences of the ideas. There are no immediate practical consequences, such as new tests to be run. Possible criteria are of relevance to the main discussion, and of validity of argument.

8.2.1 Plan of Chapter

A key point for evaluation is the design of metrics. There are currently no metrics for LOLITA-like systems[1]. The intention is to investigate the assumption (see section 1.4.4) that the value of the general NL capability of such systems is measurable. In particular, the value of the analysis capabilities will be considered, as this is usually the main factor in performance.

Whilst doing so, aspects of NL systems and of NL tasks in the general case will be discussed. The 'base' case of task-specific NL systems is considered in the light of these aspects, before considering the case of general systems. The chapter ends with conclusions.

For convenience, we talk of 'evaluation': the sense intended is the one of determining the value of something. Hence, it includes the technical senses of evaluation and assessment.

8.3 The Structure of NL Systems

NL systems do not work by magic, so we need not evaluate them as pure black boxes. On the other hand, there is little commonality of design on which to base "glass box" tests. But there are some facts about their structure which are true for all non-trivial NL systems. Some of these, we believe, are relevant to evaluation, especially in the interpretation of results.

Briefly, a system, in particular the analysis phase, is viewed as a collection of rules for converting text to a representation of the text, as required by the task being performed. It will be expected to do this on unrestricted text. This is now discussed in more detail:

8.3.1 Representation

This is the form used to express the information in the text, after analysis. We shall not include such end representations in our investigation. For one thing, the design of these representations is more a matter of Knowledge Representation – a large field in its own right. A system will clearly have problems if its representation is insufficient for the task(s). So, we assume competence for the design of representations. Defining the rules to *fill* this representation is a much harder problem.

Quite often, systems have several stages of calculation, hence a series of representations. These may not be so carefully designed. They may include 'housekeeping' information, for example to handle partial rule application or under-specification. Some form of discourse context is often an important part of the intermediate representation. We do not consider these either, apart from noting that their use will increase the complexity and/or number of rules.

8.3.2 Rules

These indicate steps in the transformation. They may convert from one representation to another value in the same representation, or to the next in the series. We do not place conditions on the form or number of the rules. Our point is to observe that a system contains some, and that they are applied in a non-trivial way. The result of analysis depends on the rules in the system. We see provision of rules as the hardest part of building an NL system.

We discuss some typical characteristics of rules in an NL system. A piece of the end result will depend on the application of a series of rules. Being correct in that piece depends on the

[1] For convenience in the discussion, we will refer to them as "general systems".

correct use of the aspects of the rules used which are relevant to that end result. Pragmatically, we do not require the complete application of a rule to be correct. That would only be required if the end result implied a need. Thus, we allow some error in the result of rule application, as long as it does not produce task errors.

In a series of rule applications, later rules may have preconditions which detect conditions set by the application of earlier rules. Some form of context may be present as 'state', and may influence rule application. This idea of state may not be important, as a particular value of state is a consequence of preceeding input. So, a rule using state could be seen as a larger rule which takes part of the previous input into account.

Sometimes, very specific rules will be required, maybe covering particular words in certain contexts. A system may be very sensitive to the presence or absence of such rules. Gaps in rules may be handled by some "backup strategies", such as chosen defaults for some condition.

The data in a system, such as portions of a general lexicon, can be viewed as rules. The distinction between such data and rules is not important in the discussion: the problems with rules will apply to data items.

All non-trivial sets of rules will be imperfect: there will be cases missing, errors in other cases, conflict between rules &c. Errors at one level of rules may have adverse consequences in later stages, eg by falling foul of preconditions. Problems can propagate from one level to the next. The case where several errors cancel each other to produce correct output is also possible.

8.3.3 Architecture

System architecture does not appear very relevant in this argument: one may view it as the *method* of rule application. For example, a use of under-specification means keeping options open on which exact rules to apply. Alternatively, it represents the construction of more complex rules from simple rules.

We assume that the mechanisms of rule application are a minor concern compared to the rules themselves.

Trainable systems are seen as cases where the rules are generated automatically, rather than being written by hand.

8.3.4 Comprehending an NL System

The obvious question to ask is, what properties does this collection of rules have? How do we *understand* the overall system? This is one sense of evaluation, but not the main sense of our discussion, of how useful such an artefact is. The questions of this section are more of a technical nature, about understanding what we have created.

The rules essentially cover many cases, so there is much detail to check. The details will typically be too much for a single person to comprehend at once. Consider the difficulty of maintaining a large grammar. Furthermore, one must consider how *combinations* of rules interact, in sequence and in parallel - if several can apply at any one time. Matching of effects (post-conditions) and pre-conditions will be complex. A big part of understanding a system is also in finding its weaknesses.

Correctness of individual rules can be judged by examination, but this is no guarantee of how the rules will behave in combination. Again, grammar writing is a good example. Completeness of the rule set is hard to establish: we do not have methods of ensuring every facet of language is captured. Note that correctness and completeness only with respect to producing correct output are required – a pragmatic view.

There is an operational aspect: we can run the system on pieces of text and see what happens. But how do we interpret the results? There is no general categorisation of NL, so we cannot vary conditions and fix certain aspects of variables of NL in a principled way: there are no obvious 'knobs' to turn. In particular, such inductive techniques assume that a fairly smooth 'function'

is generating the observations, for example some regular relation between quantities in a physical experiment. We cannot be sure of this general uniformity in the behaviour of collections of rules. Hence we cannot construct useful statements about more general behaviour from this evidence, and this evidence will remain anecdotal. Of course, developers can use it heuristically in their work, for example as a confirmation that strategies work in certain cases.

Finally, we note that many MUC-6 competitors, including the LNLE, emphasised the importance of fast testing times. This suggests a reliance on a prototyping approach, which we see as a consequence of the operational interpretation of a system's rules being the preferred way of understanding them.

8.3.5 Development

The underlying motivation is to improve the set of rules. There are several actions that can occur:

- Adding new rules to the system, for phenomena that are not in coverage. Such changes are not monotonic - conflicts and interference with existing rules can occur.
- Generalising rules. There are practical limits to the rules in a system. A large number of them will occupy much space, and selection of appropriate rules during analysis will be time-consuming. Hence, some compaction is necessary.

  This is sometimes problematic, eg by over-generalising a rule, which allows it to match more cases than intended. Grammar writing is a good example of this.
- Correcting rules. The importance for understanding a system of finding its weaknesses was mentioned above. The previous activities are likely to introduce flaws, so this activity is quite necessary. Note that as a software system, most rules will be easy to change, hence people will want to repair flaws soon after detection. Some rules will be easier to change than others, eg adding simple data compared to changes in the grammar.

This cycle has no natural stopping point, and in non-trivial systems, there is always plenty to do. Hence, it gives rise to a prototyping approach. As a consequence, NL systems are not "set in stone": evaluation results will not be final, and there is a possibility that a small set of changes can significantly improve (or change) performance.

Discussion

A view of NL systems as a collection of rules has been presented. We have observed that in non-trivial systems, there will be missing rules, conflicts between rules, and errors in rules. Note that NL is complex, and few simple rules exist. Hence, a system handling NL will need many rules.

The detail present in a set of rules is such that understanding them takes a large amount of effort. Thus, the usual way of understanding a system is through running it on some text, and making judgements on the results. This is mainly anecdotal evidence, however, as good performance on one piece of text is no guarantee of the same on another: a complex set of discrete rules will not always provide the homogeneity of behaviour which makes such induction plausible.

One sense of evaluation is to understand the properties of this set of rules. Alternatively, since the set of rules is unlikely to be perfect, evaluation can be seen as determining the *strengths* and the *weaknesses* of this set, and then to understand what the set of rules can be used for. Such an exercise usually refers to a task or purpose.

Why has this model been introduced? No strong scientific consequence such as prediction of some new phenomena has been derived from the model. But it does have weaker scientific value as a *description* of NL systems: it explains some aspects of such systems, such as the way they are commonly understood (operationally), and the way systems are developed.

8.4 The Structure of NL Tasks

We examine the notions involved in NL tasks: the kinds of task, how a task is defined, and how a human can be rated on an NL task.

8.4.1 Kinds of Task

We can distinguish several broad classes of NL task:

- Those which humans already do, and for which notions of success exist. An example is translation. However, the intuitive ideas of performance are found hard to convert into quantifiable tests. There are also high expectations for performance, since humans do the tasks to a high standard.
- Those which humans could do, but are tedious or repetitive. Information Extraction on a large scale, ie on many articles, is an example. Some intuitive standards may exist, but they will not have the history of the kind above, so will be less developed. Exact quantification may be difficult. Since humans can perform the task well in small batches, expectation may be high.
- Those which have become possible by the new technology, for example, NL interfaces to various tools. In these, new standards need to be devised.

The main point is, standards for tasks vary widely, and for some kinds of tasks, the standards may lack a strong foundation. In all cases, detailed quantification is not easy.

8.4.2 Defining a Task

An NL task does something with NL text, so one can view a task as a *function* from input text to result. We lack means to describe, or to *specify*, this function exactly, since we lack means of talking about NL precisely. One reason for this is the sheer variability of NL: there are many ways to express a given statement. So, task definition is in some form of NL, which is an attempt to describe this function.

Note that assuming a function introduces the notion of function domain. To avoid confusion with an informal term, we call it the "task-function domain". This will be the space that the function operates on. We only note the task-function domain's existence; we do not need to know its structure.

8.4.3 Human Performance

It is interesting to discuss the evaluation of human performance before we discuss machine performance. Consider some kind of intelligence analyst being assigned to a new task.

We are unlikely to present the analyst with a detailed explanation of a task. More likely is a short overview of the task at a high level, relying on the analyst's experience and judgement for the finer details. We would not perform a detailed evaluation of their performance. Their qualifications, which we discuss below, will indicate a certain level of competence, but for the remainder, again we rely on their experience.

Examinations are an interesting case. They are effectively a sampling of a person's performance at some task. Inductively, this says little about the person's overall performance, but we may make assumptions about the reasons for the person's performance in an exam. In preparing for an exam, the person would have studied all of the areas their tutor required, any of which could have appeared in the exam. Note that the tutor has control over the curriculum and the examination questions. They also decide the metrics, or the marking scheme. Unless the person was "question spotting", the sample performance can be taken as representative of overall performance.

Humans tend to make particular mistakes. Cognitive limits are one source, eg memory limits leading to a failure to combine scattered information. Fatigue is another source, due to the repetitiveness of a task &c. One would expect that the slips are minor, with no major errors, unless the relevant information is unclear. In cases of uncertainty, humans can draw attention to the problems and let other more qualified people decide on the result.

A frequently used technique for marking is to assume maximum marks for some answer, and to deduct marks for mistakes. In the author's experience, this is used in marking language exams, where marks are subtracted for errors in forming tenses or cases. This avoids having to allocate marks for the correct portions of an answer.

8.5 Machine Performance on Single Tasks

An implemented system can be seen as a function from input text to result, one that attempts to mirror a task function. The task function cannot be captured directly, hence the machine version will be an *approximation* to the task function. The point for evaluation is how good the approximation is. But how to determine this?

The conditions of NL and systems has suggested an operational understanding of performance of a task - of how a system behaves. Thus, data will be obtained by running systems on a sample of texts. The problem is now how to assess the approximation from a sample of performance. Relevant issues are discussed in the subsections.

8.5.1 Value of Performance

Intuitively, there are reasons for requiring a task to be done. The results will have some value to someone. Correspondingly, value will be lost if the task is done imperfectly. How do we quantify task performance? For example, how do we allocate scores to the success or failure of any part of a task? Are failures more significant than successes? These are hard questions.

MUC-6 allocates one point per feature in the answer. This was argued against in section 6.2.5: this method allows systems to pick up scores for the easier parts of a task, without doing well on the harder parts, which may carry more information and hence more value. Use of correctness analysis was suggested, to allocate more importance to the parts of a task on which all systems do less well. This still ignores the semantic content of the output of a task.

Eventually, clients must be involved in characterising value. Developers are not well placed to do this alone. For example, what developers think is hard or interesting may be irrelevant to a user. Consequently, developer-designed evaluations may end up too artificial and irrelevant to real applications. Clients are better able to indicate the worth of a system; one way of doing this is by indicating the financial consequences of the information. For example, incorrect answers could be much more significant than successes or omissions.

We can still discuss some issues concerned with measures, to help clients understand NL systems and their particular strengths and weaknesses, in order to decide what they want. Developers should also investigate clients' views.

8.5.2 Machine Errors

The discussion of value above did not consider the source of the task output. Uncontroversially, machines will make different mistakes to humans. Also, the limitations of human performance will not apply to machines. So, ideas of value should take these into account. For example, basic string matching is not as valuable as complex inference.

What kinds of mistakes will systems make? We cannot answer this here. No serious study has been carried out by us, and we are not aware of any in the literature. One difficulty is that mistakes will have to be defined relative to a task, and there is not much data available from

several systems on the same task. The MUC-6 response files may be useful for such a study, and the author's template comparison tool will ease analysis of the responses.

8.5.3 Forms of Judgement

There are certain questions that are sensible to ask in evaluation, and so evaluation should be oriented to answering them. Consequently, some questions may sound reasonable but may not serve the purposes of evaluation. Pragmatically, there are only two questions that need to be asked:

- "Is system A better than system B?" This is asked when two or more systems are being considered for use in a task, and only one is required.
- "Is system A good enough for use?" This is asked when deciding if deployment of system A is worthwhile.

Note that there is a dependence between the two: comparison may not be useful if neither system is good enough. One may also ask questions like:

- "Is system A 20% better than system B?"
- "How good is system A?"

But these do not seem as central to the concerns of evaluation as the first questions: they have more academic interest. Note that the form of these questions implies precision or quantification.

The question used in MUC-6 is of whether the difference in performance of a pair of systems is statistically significant. This is a weaker question than any above. However, the inputs to the statistical tests are precise numbers. A detailed index of performance is calculated, then reduced to a set of judgements of significance against other systems. The meaning of the numerical scale is not clear, as discussed in various places in chapter 6. Officially, there is not much importance attached to the original numbers. Informal comparison cannot be prevented though.

The point of this section is that detailed quantifications of performance may be hard to justify and hard to understand. For example, what is the difference between a score of n and of $n + 1$? At what point $(n + i)$ does the difference become important? If the important questions can be answered straightforwardly, then high detail is superfluous. There is no automatic need to calculate a precise answer. Detail should only be used where needed to answer the questions asked.

Different techniques may be needed to answer the two questions. The question of relative performance is examined next. The question of adequacy would seem to depend on some notion of value. As noted above, this will require input from users of NL systems.

8.5.4 Relative Performance

This is the first of our main questions in evaluation. In what sense may a system be better than another? We consider factors in evaluating relative performance.

A big part of deciding if one system is better than another is to decide whether the value of one system is more than the value of the other. This requires an assignment of *value* to parts of an answer, which was discussed in section 8.5.1. In this section, we consider the consequences of comparing two systems, and show how this affects an assignment of value.

Imagine two sets A and B, representing the output of two systems, and assume that we can compare A and B to identify similar behaviour of output. A third set C contains the correct output; for the moment, C is a model answer. Assume that we can also check A and B against C, to determine where the systems are correct, and where they are not. We are only interested here in the *binary judgement* of whether a system is correct for a particular part of the answer.

Note that this is *not* the same as scoring in the MUC sense: for the correctness judgements to become scores, a value must be attached to them.

The intersections of these sets produce seven possible regions:

- C(AB): where both A and B are correct.
- C(A): where only A is correct.
- C(B): where only B is correct.
- C($\emptyset$): output which both A and B under-generate.
- C'(AB): the common output of A and B which is incorrect.
- C'(A): the output of A where it alone is incorrect.
- C'(B): the output of B where it alone is incorrect.

Observe that dependencies exist between the sets. For example, regardless of A and B, the size of C is fixed. The performances of A and B partition C. What is not in C(AB) must appear in one of the three other partitions. For example, if C(A) and C(B) are both large subsets of C-C($\emptyset$)[2], then C(AB) must be small. In other words, if both systems get many different parts of a task correct, then the amount of overlap or similarity will be small.

How does this model reflect on comparisons? The content of C(AB) can be ignored, since on this portion, A and B are indistinguishable. The same applies to C($\emptyset$) and C'(AB). Therefore, we do not need to assign value to these parts of the performance, since the value for both A and B will be indistinguishable.

The interesting points are where the systems disagree. Our value judgement is thus a consideration of the remaining sets C(A), C(B), C'(A), and C'(B). We suggest that the judgement "A is better than B" can be understood as C(A) having greater value than C(B), and the loss of value due to C'(A) being less than the loss due to C'(B), according to some valuation of task performance.

Informally, this means B performs almost as well as A, but A's mistakes are not as serious. To decide this judgement, we need to consider the *value* of C(A) against C(B), and the *value* of C'(A) against C'(B). There are two main cases: where A and B are quite similar, and when they are not.

Comparing Similar Systems

It is easier to compare similar things than to compare dissimilar things, as the similarities can be ignored and focus lies on the particular differences.

A is similar to B if the intersection of their performance is much bigger than the performance outside the intersection. Also assume that C(AB) is a major component of C-C($\emptyset$), ie that both systems get most things correct: this ensures that C(A) and C(B) are relatively small. These conditions are not intended as exact. One could use the size of a set, ie the number of components of an answer, as a guide. (This is different from the 'value' of a set, which is the value of the output it contains.)

Under our assumptions, C(A) and C(B) will be small compared to C(AB). Hence, to decide this part of the judgement, one only needs to decide the value of a small part of the correct answer.

The difference between C'(B) and C'(A), the errors that systems make, is not as restricted, so there may be more to check. It is not certain whether the valuation of errors should be similar to the valuation of successes, so different techniques may apply when comparing the sets of incorrect output. Eg, some mistakes could be so serious as to cancel out an otherwise good performance.

[2]The portion of C which A or B produce.

One possible scenario is that when comparing two systems, one looks at C'(A) and C'(B) first, and if they are bad enough, we do not even consider C(A) against C(B). We noted above that no study has been made of the errors that machines make. More research is needed to investigate the demands of this part of the comparison.

Comparing Less Similar Systems

Here, C(AB) will be relatively small, with the consequence that C(A) and C(B) are much larger. The situation for comparing C'(A) and C'(B) will be similar to the previous case, so we concentrate on comparing the values of C(A) and C(B).

For this, we must compare the value of two large and dissimilar sets, which is clearly harder than making a valuation of two small sets. It is possible that too big a difference in performance may render meaningless the question of which is better. This is because we cannot provide a good reason to choose between the good points of one system, vs. the (different) good points of another system.

A pragmatic question may be asked: how frequent is the case of comparing systems that do not show great similarity? Again, this has not been studied by anyone. An answer may be possible using the author's similarity analysis method (see section 5.7), but not immediately: more work is required to process the raw figures. An informal test on the MUC-6 results for TE found only one case where both C(A) and C(B) were larger than 1/3 of the size of C(AB). All other cases showed either a very large C(AB) or a big difference between the sizes of C(A) and C(B). This suggests that in TE the case of comparing highly dissimilar systems is not common.

Discussion

Initially, a distinction was made between whether something is correct or not, and the value of this condition. The former is a binary decision, and can be produced automatically, whereas the latter is more subjective. One could avoid the issue by awarding one point for each correct answer, but it is clear that different parts of an answer have differing values. Hence value should be assigned more carefully.

We have suggested that a judgement of whether one system is better than another can be decided by a comparison of distinct correct performance (C(A) against C(B)), and a comparison of distinct incorrect performance (C'(A) against C'(B)). The other sets (C(AB), C'(AB), C($\emptyset$)) can be ignored, since the systems are indistinguishable on them.

This means that we only need consider the value of these four sets, which has several consequences:

- We do not need to specify or justify value in detail for all parts of a task, just the ones where a notion of value is necessary.
- So, we do not need to justify the value of parts of the task which are simple enough for both systems to get right, or the parts which are too hard for either.
- Correct performance has been separated from incorrect performance. Different methods of valuation may be required for each, eg some errors can make an otherwise good performance unacceptable.
- The important factors in a comparison are automatically identifiable and software tools can be produced to aid in the comparison. It also exposes the reasons for making a judgement.

So far, there has been no discussion of *how* we establish the value. It is suggested that value be decided *after* examining the four important sets. Initially this will require a human judge, but as experience is gained - especially in a series of comparisons, eg for successive versions of a system, some decisions on value can be characterised and automated. One objection is that this approach increases workload for evaluation. The decisions may not always require a detailed comparison;

for example, a brief examination of the key points of the task (from a user's perspective) could provide an adequate decision.

We note that this approach to comparing two systems has never been implemented, so there is no empirical evidence for or against it. But the degree of similarity between systems in MUC-6 suggests that it is a worthwhile subject for further research. Furthermore, there is *no evidence* about why one system should be preferred to another, hence a study of this kind would have several benefits.

To conclude, there is a difference between the notion of being correct, and the value of being correct or not. The former can be automatically decided, but the hard decisions on what the differences mean are ideally made with human assistance. In comparing two systems, there are essentially two cases to consider, a comparison of where systems differ but are correct, and a comparison of where they differ and are incorrect. The remaining parts of performance can be ignored, since the systems cannot be distinguished on these parts. It is possible that such comparisons can be made by a human judge with little work, especially if the differences are few, or if one system makes serious mistakes; suitable software tools will also help in these decisions.

However, there is a general lack of evidence either for or against this idea, so further research is required. In particular, we are not aware of any work in the field which examines why people may prefer one system to another.

8.5.5 The 80-20 Problem

This is the situation of 80% of functionality being provided by 20% effort. Alternatively, of 20% of functionality being provided by 80% effort. This problem is well known in NL work: basic rules give good performance, but improvements in performance become increasingly more difficult to achieve. It appears unavoidable, so evaluation may need to take it into account. This works both ways: to avoid optimistic figures in the easy stages, and to avoid pessimistic figures in the harder ones. It also admits that perfection in NL systems is practically infeasible, and is not a realistic target for work.

8.5.6 Changes in Performance

One characteristic of NL systems is that they can be easily changed. Thus, evaluation results only apply to a system at a certain time, and even small changes can produce a large difference in result. The ability to change is important for NL systems, and should be considered in evaluation. Other literature takes a static view of things, but a *dynamic* view seems more accurate. Understanding the changes in performance is an important part of development work.

There are several issues to discuss:

- How change is tested.
- How often change is tested.
- Valuation of the change in performance.

Our main interest is in valuation. An important point is that changes may not always be improvements. Furthermore that losses in performance can reduce the effect of big improvements, to show only a mild improvement. One can score the two performances and compare numbers, maybe comparing numbers from the same parts of task (eg performance on a certain slot). But, comparing a system to a changed self is a special case of system comparison (see section 8.5.4) – in particular, of the more favourable case where similarity is high. In this model, value is only considered for the qualitative differences in performance.

The other questions are difficult, and involve issues outside technical detail. In particular, they do not affect the value of a system: they affect the process of building a system, such as how fast the process proceeds and how general the results are.

On how often, there is no limit: the decision would seem to involve a tradeoff of thoroughness, informativeness, and cost of evaluation. A system can be tested after every change, but this will be expensive, and the changes in performance could be minor – eg affecting only a few cases in the test set. Infrequent testing will provide a global picture, but the results will be harder to interpret.

The question of how the re-testing is implemented involves issues of sound experimental design. For example, intensive development on a single small test set may produce a system tuned to that set alone, with poor performance on new texts. Blind testing has its faults too: if no information about common mistakes is available to developers, then they may be optimising to the scoring metric. One MUC-6 competitor used a strategy which combines the two approaches, working with two test sets (open and blind) – this is a good idea. Comparing the changes in result on an open set with those on a blind set may be an interesting project.

8.6 General Systems

The purpose of this section is to investigate the notion of general systems in the framework outlined so far, of viewing a system as a set of rules which approximates an NL task function, and of viewing value pragmatically. In particular, this framework does not allow evaluation without a task: we have no notion of *directly* evaluating the 'core' of the general system. The framework also gives us some tools to analyse the idea of general systems. We consider what a general system is, then whether it can be evaluated inside the framework.

8.6.1 Definition

General systems, as we call them, are usually defined as systems that can be used to implement various tasks in different domains, when suitable resources are added. For example, to do template-based Information Extraction on financial articles requires addition of a module which produces templates from internal analysis results, plus domain-specific knowledge on finance. We use the term 'application' for such a customised system. LOLITA is the prime example of a general system in this discussion. Another example is CLARE [Alshawi *et al.*, 1992].

This definition is vague. The author understands it as an intuitive notion: people assume that a 'general' NL core can be constructed, and its use in different applications is simply a matter of supplying appropriate resources. We say 'assume' because there is no proof that such an item can be constructed.

Customisation

Customisation is a key point for general systems. What form does it take? Descriptions of LOLITA and CLARE both imply that customisation is a matter of adding new words to the lexicons and incorporating application-specific knowledge at the pragmatics stage and above. There is no fixed process for customisation. It may be said with fair certainty that no general system is good enough for customisation to be as simple as this. For example, much work on LOLITA for MUC-6 was at the syntax and semantics levels.

This means that general systems are still "in development". Any evaluation of applications will be evaluating both an imperfect core and the customisations to it. It may not be possible to separate the results of either. It may also be hard to distinguish customisations from core in the code: ie, the author expects customisations at the levels of syntax and semantics to be necessary, as well as the kinds mentioned above.

When a change is required in the system, it may be unclear whether it is a customisation for the application, or a more general change. Heuristics which work on financial newspaper articles are a good example: they may be useful in similar domains. Keeping such customisations as very application-specific could be wasteful, as similar applications will need identical rules. Hence, the preference for adding them to the core. The reverse is also problematic. If anything that looks

task-specific is added as a customisation to be used by just one application, then the core will end up being trivial in comparison. The hard work will be done in the 'customised' portion.

So, there are no clear limits on customisations. Without these limits, one can talk of the core as being customised to conditions like a sentence containing 'the': obviously, something that should be implemented in the core. Also, the idea of a core system loses coherence: it cannot be distinguished from a loose toolkit of NL components, where a subset of the components can be bundled together to form an application. In such a toolkit, there is no coherent whole that can be a subject for evaluation.

Development

How are general systems developed? One possibility is by a developer's notion of what a task-independent language processor should look like. However, as noted before, this is a weak notion: there is no general description of NL which can be used as a guide for development. This system would still need applications in order to be tested.

Development seems dependent on applications, so the expected model is by prototyping on one or more applications. There is no fixed process for customising a system, so customisation will be informal. Applications can be evaluated as if they were single task systems. Unfortunately, changes during work on one application may affect performance on other applications, and the result is not always predictable in a complex system. This problem is increased if the distinction between task customisations and core is not clear, because customisations may clash.

Thus, development seems more complex, if good performance is required from all tasks – which is a reasonable requirement. In other words, work for the nth application must preserve performance for the other $n-1$ applications. This will produce increasing amounts of work, especially in regression testing. Repairing degradations in other applications may also be a complex piece of software maintenance.

Summary

The idea of a general system seems very intuitive. The only reasonable way of developing one is through applications. No system can be called general: development of applications must therefore include development of the underlying core, and likewise for evaluation. Possible customisations are not restricted; this raises doubts about the reality of a core system. In one extreme, much of the processing could be done by customisations; in another, the core is indistinguishable from a loose toolkit of NL components.

8.6.2 Evaluating General Systems

The question is, can we assign value to a general system in this framework? We have no criteria of linguistic correctness, so we cannot value it on how good a model of language it is. The real task for general systems is to help implement applications quickly and easily, and for them to have high value. This is definitely a point of economic value, in contrast to the functionality value we've been considering so far. Thus, general systems do not have value in the framework.

We note that if it did, this would assign value to a *component* of an NL system – for what is a general system but a component in a larger application? – and this would contradict the earlier step of ignoring components like parsers in conventional systems. In a sense, the question of evaluating general systems on functionality is meaningless.

Implications for Performance

How good will the applications built from general systems be? We are trading specificity of implementation with supposed economic gains – that later applications will be cheaper to build, so the performances of dedicated systems cannot be expected. Bearing this in mind, should one

be more lenient in interpreting the differences between the two kinds of system? The answer is no: the dedicated system will only be less valuable if the economic advantages of the application using the general system outweigh its weaknesses in performance.

8.7 Conclusions

This chapter has been an informal discussion of aspects of NL systems and NL tasks, and the consequences of combining theses aspects, which the author believes should play a part in evaluation work. The evaluation of general systems has been discussed in the light of these points. The main points of this chapter are:

- NL systems have been modelled as an algorithm for converting text to some logical representation by application of a set of rules. This has several consequences:
  - The rules determine the performance of the system.
  - A system will contain many rules, and there will be omissions and conflicts in a large set of rules.
  - The complexity of rules in a non-trivial system prohibits comprehension; understanding is usually gained through executing the rules.
  - Typical development consists of adding and debugging rules in a prototyping fashion.
  - The rules are easily modified; hence the system's performance can change significantly in a short space of time.
- NL tasks have been characterised as a function. Task definitions are attempts in NL to describe these functions. There are several kinds of task, whose standards depend on how well humans do them, and how natural the task is. Humans make certain kinds of mistakes, such as through tiredness.
- An NL system is an approximation, through a set of rules, to an NL task function. Evaluation involves determining the value of the approximation.
  - Value of output should be rated by those using the output. A developer's notion of value may be artificial. The value of successes and of mistakes need not be the same.
  - Machines do not make the mistakes humans make. Hence, human standards should not be applied. There is a need to study the kinds of mistakes machines do make: the MUC-6 responses may be suitable.
  - There are two essential questions in evaluation:
    * "Is system A better than system B?"
    * "Is system A good enough for use?"

    Other questions are academic, and could involve attempts to quantify performance beyond what can be justified or understood. The detailed quantification used in MUC-6 is too powerful. Furthermore, it is used inconsistently: the official results are binary statements of statistically significant difference, a big reduction in detail. Evaluation should concentrate on the important questions. The techniques needed to answer them may be different.
  - When comparing two systems, there will be some behaviour common to both. So to decide which is better, we can ignore the similarities and concentrate on the differences. In particular, there are two cases: comparing the value of what the systems uniquely get correct (ie, that the other doesn't), and comparing the value of what they uniquely get wrong. Value is hard to define, but this approach limits the parts of a task to which value *need* be assigned to that required to decide the above cases. It is quite possible that a human judge can (with suitable tool support) decide the cases with little effort, but more research is required on this. In particular, there is a lack in the field for research on *why* we might prefer one system to another.

  - NL suffers from the 80-20 problem: good performance is possible with some basic techniques, but improving the performance gets increasingly difficult. Acknowledgement of the problem will mean adoption of more appropriate targets for work. Perfect systems may be impossible.
  - NL systems can be modified easily, with various effects on performance. Evaluating changes is a special case of comparative evaluation.

- There is no precise definition of a general system, which leaves the concept in a weak position. For example, what is the difference between a general system and a loose toolkit of NL components? In our framework, general systems have no value: the only real task they have is in implementing other applications, which themselves may have value. Thus their 'value' is economic, although proof of this value has not been demonstrated for any such system.

The intention of this chapter was to outline a developer's view of evaluation, which considers the software aspects of NL systems and a pragmatic view of the value of NL systems. This has been achieved.

The original question, of how to evaluate LOLITA, has not been closed. In our framework, we can not assign a value to such systems, only to the applications they implement. Any value LOLITA has is primarily economic, though proof is required that use of LOLITA does result in cheaper development. Some other approach may be required for analysing general systems. However, the discussion on single task evaluation should improve the quality of evaluations for LOLITA-based applications.

Chapter 9

Conclusions

This chapter considers how well the work in this thesis has satisfied the problem-specific criteria, and the methodological criteria. The particular successes of the work are listed, and suggestions are made for future research.

9.1 Fulfilment of the Project Aims

We discuss how well the aims of chapter 2 are satisfied, with reference to the methodological criteria of section 1.2.

9.1.1 Aim 1.1: LOLITA in MUC-6

- Description of workings of LOLITA: LOLITA has been described in some detail, and information given about the adaption of LOLITA for the MUC-6 tasks. We have also discussed the use of Haskell, considering the advantages and disadvantages. This aim is satisfied in chapter 3, and in the performance analysis of chapter 4.

- Detailed analysis of performance: a single article (see Appendix A), chosen from the mid-range of LOLITA performance, has been analysed in detail in chapter 4. We have also compared LOLITA performance on this article to the other competitors in MUC-6. A software tool has been developed to aid such comparisons.

  The overall scores are analysed in chapter 5. A methodology for analysing scores is discussed. Two novel methods of analysis are developed for comparing system performance, and are applied to all tasks. One of these, "correctness analysis", shows particular promise: it provides a characterisation of task difficulty, and shows how systems attack a task.

  In all analyses, the current performance and its relation to the original (evaluation) performance is discussed. The detail of these several analyses satisfy the aim.

- Relevance of MUC-6 results to LOLITA: this aim is harder to assess, as firm criteria could not be established for it. Methods of interpreting the results have been discussed in chapters 4, 5, and 6: these include weighting of scores to highlight the important or difficult parts of a task, elimination of parts of a task which can be achieved using simple text matching, transformation of the text to remove the cues which provide simple-to-get marks.

  The fact remains, that LOLITA scores are low, however one interprets the results. We have evidence, through correctness analysis, that LOLITA makes many mistakes on the features of a task which most other systems get correct, which leads to a belief that marks can be gained in this area more successfully than other systems can gain marks on the more difficult parts of a task.

9.1.2 Aim 1.2 : Analysis of MUC-6

This aim is covered by chapter 6. We have considered the following:

- The goals of MUC-6: these are briefly discussed, at the start of the chapter, with reference to [Grishman and Sundheim, 1995]. That paper argues that most of the goals are satisfied, and there are few reasons to disagree. Our major criticism is of how MUC-6 encouraged "deeper understanding" in a task: there is no evidence that the scores obtained on, say, the Coreference task could not be obtained by mainly surface techniques and heuristics.
- The difficulties in defining tasks. We note a few inconsistencies and possible weaknesses, eg in interpreting ST output.
- Form of representation for answers – particularly for the template tasks, where we suggested a finer grain of representation and the basing of templates on specific strings from the article (ie, annotated with their position). It was also argued that a single, multi-purpose template design was not necessary: a basic logical representation from which human-oriented views could be derived, may be more productive.
- Consistency between ST and TE task keys: although the keys for these overlap, it appears that the identicality of content for these was not checked. The difference was fortunately small.
- The scoring algorithm: the basic details are considered. Some problems of template alignment may be reduced by annotating strings with their position in the text. A special treatment for IN_AND_OUT templates in the key was noticed; this would be useful for response templates too, in reducing mismapping for systems that produce under-filled templates with similar content.
- Several methods of weighting the final scores were discussed. This would emphasise the more important or the more difficult aspects of a task.
- Correctness analysis is used to provide another view of the tasks and of how systems approached them (see section 5.6).

And the following changes were suggested:

- Some of the TE templates appear in ST output, so in a sense they have already been scored. A different view of ST performance can be gained by omitting these templates from the final scores. Several points of precision are lost by all systems under this procedure, and most of the systems lose two or more points of recall.
- From the observation that some parts of a task are trivial, we discuss ways of objectively eliminating such parts from the task results. This provides a baseline of performance, which separates 'real' NL processing from simple text manipulation.
- Inspired by the scoring methods of TREC-3, and the observation that most systems will agree on some parts of an NL task, it is argued that the output of competing systems can be used to make preparation of answer keys more efficient. The result of scoring with these keys will not be an absolute value, since all systems will miss some answers; instead, it will be relative, showing how systems did relative to each other. A rough method is outlined, but needs more research.
- Providing texts and keys is expensive, so it makes sense to make full use of them. A number of transformations is discussed, which make the task more difficult in a predictable way. A group of these, based on modification of cues in the text, appears useful. Transformations which use information in the keys, such as coreference information, appear less useful.

The discussion has kept to the technical side of MUC-6. Wider aspects – including the political, economic, and social – are difficult to analyse. Answers to them depend, as [Galliers and Sparck Jones, 1993] noted, on the motivation for evaluating; these motivations are not explicitly stated for MUC-6. On the subject of competitions, we note that much can be gained from qualitative comparisons of performance, both to examine the nature of a task, and the similarities and differences between systems.

As noted in the criteria, this aim is open-ended. The author believes this aim is satisfied by the depth of analysis, and the novelty of his suggestions.

9.1.3 Aim 2: Investigating the Evaluation of General NL Systems

This is covered by chapters 7 and 8.

- Literature Review: literature relevant to this aim has been studied, and interesting ideas noted. There is very little concerned with general systems, and this small amount is not encouraging. For example, [Galliers and Sparck Jones, 1993] concludes that systems can only be properly evaluated on single tasks, but that performance on a single task is no guarantee of performance on other tasks.
- Theoretical Discussion: a developer's view of evaluation is given. The consequences of NL systems as software of a particular kind, and of a pragmatic valuation of task performance, are discussed in detail. A particular point is the need to study the kinds of errors that machines make; MUC-6 responses may be a suitable corpus for this. General NL systems are briefly considered. It is concluded that their 'core' linguistic capability cannot be evaluated within the framework; their main worth is economic. Thus, LOLITA itself cannot be evaluated, but the points raised in this chapter may lead to better evaluations of its applications.

Again, this aim was open-ended, and objective criteria could not be identified. Several ideals were suggested, such as examining the basic notions of evaluating general NL systems, and of making a contribution to the field. The discussion has been from "first principles", and introduces several novel aspects into the debate. The author believes this aim is reasonably satisfied.

9.1.4 Satisfaction of Methodological Criteria

Empirical Part

The main criteria were thoroughness, and the requirement that suggestions for improvements in the MUC form were feasible and well-justified. The author has analysed LOLITA performance and the MUC-6 format in detail, so the former has been satisfied. Several interesting suggestions were made for extending MUC, most of which are implementable now, the remainder needing further research.

Theoretical Part

The main criteria here were of making a contribution to the problem's solution, justifying the ideas under a neo-pragmatic view of science, and suggesting implementable schemes for evaluating general NL systems. The question of how to evaluate NL systems remains open, but the thesis author believes important and relevant issues have been raised.

9.2 Successes of the Project

9.2.1 Practical Aspects

- Up-to-date description of LOLITA, relevant to MUC-6.

- The author has also made many big improvements to the LOLITA system, resulting in improved performance, especially in its speed.
- Detailed analysis of overall performance in MUC-6.
- Detailed analysis of performance on a single article.
- Analysis of what tasks require on a single article. That is, the simplest steps required to get an answer correct are discussed for most features in all four tasks. This is particularly novel for MUC-6.
- Implementation of a tool to help investigation of task performance. The tool also allows qualitative comparison between different systems, or between versions of the same system.
- Use of this tool to implement "correctness analysis" (see section 5.6), providing an alternative view of MUC-6 results from several systems. It also characterises task difficulty.
- Use of this tool to analyse similarity of behaviour for correctness, under-generation, &c. This shows how many times a pair of systems do the same thing (see section 5.7).
- Suggestions for MUC-style evaluations. These were listed in section 9.1.2.

9.2.2 Theoretical Aspects

- An up-to-date literature review.
- A "first principles" investigation of evaluation, from the viewpoint of NL systems as software of a particular kind, using a pragmatic notion of value.

9.3 Future Work

There is much to choose from, so below are listed the suggestions which the author believes are most interesting.

Empirical Part

- Implementation of the suggestions made for MUC-style evaluations, particularly the use of positions for string fills and the alternative template design.
- Further analysis of comparative performance for the MUC-6 competitors. This should be more interesting for ST and TE than NE. More work is needed if the method is to be used for Coreference.
- More research on the "automatic assistance for annotation" idea.
- For LOLITA, attack on the errors that most systems get correct. These are expected to be relatively simple to correct.

Theoretical Part

There is still much to do with evaluation. Debate is needed for the ideas suggested in chapter 8. The following practical steps will provide useful information for such future debate.

- Investigate the kinds of errors that systems make.
- Implement the ideas on relative performance.

Appendix A

The Chosen Article, 9306220057

This article was analysed in chapter 4. Permission has been kindly granted by the Linguistic Data Consortium for its reproduction here.

The keys are reproduced by permission of the MUC-6 organisers. Some editing of long lines has been done, to fit on the page.

A.1 The Article

```
<DOC>
<DOCID> wsj93_005.0011 </DOCID>
<DOCNO> 930622-0057. </DOCNO>
<HL>    Who's News:
@  Johnson & Johnson
@  Manager Makes Move
@  To Genetic Therapy </HL>
<DD> 06/22/93 </DD>
<SO> WALL STREET JOURNAL (J), PAGE B12 </SO>
<CO>    GTII JNJ </CO>
<IN> BIOTECHNOLOGY (BTC), DRUG MANUFACTURERS (DRG),
   MEDICAL & BIOLOGICAL TECHNOLOGY (MTC) </IN>
<DATELINE> GAITHERSBURG, Md. </DATELINE>
<TXT>
<p>
   Michael D. Casey, a top Johnson & Johnson
manager, moved to Genetic Therapy Inc., a small biotechnology
concern here, to become its president and chief operating officer.
</p>
<p>
   Mr. Casey, 46 years old, was president of J&J's McNeil
Pharmaceutical subsidiary, which was merged with another J&J unit,
Ortho Pharmaceutical Corp., this year in a cost-cutting move.
</p>
<p>
   Mr. Casey succeeds M. James Barrett, 50, as president of Genetic
Therapy. Mr. Barrett remains chief executive officer and becomes
chairman.
</p>
<p>
   Mr. Casey said he made the move to the smaller company because he
saw health care moving toward technologies like the company's gene
therapy products. "I believe that the field is emerging and is
prepared to break loose," he said.
</p>
<p>
   Mr. Casey declined to divulge his compensation, which he said was
comparable to his package at J&J, but acknowledged that the equity
portion was higher at Genetic Therapy.
</p>
```

```
<p>
   Noting other recent moves by pharmaceutical executives to small
biotech companies, John T.W. Hawkins, the executive recruiter who
arranged the Genetic Therapy placement, said, "The equity play is
obviously the draw for many of these executives in evaluating small
and emerging companies." But, he added, "fundamentally, it's the
excitement and challenge of building an emerging pharmaceutical
company" that attracts the executives.
</p>
</TXT>
</DOC>
```

A.2 Named Entity Key

```
<DOC>
<DOCID> wsj93_005.0011 </DOCID>
<DOCNO> 930622-0057. </DOCNO>
<HL>    Who's News:
@  <ENAMEX TYPE="ORGANIZATION">Johnson & Johnson</ENAMEX>
@  Manager Makes Move
@  To <ENAMEX TYPE="ORGANIZATION">Genetic Therapy</ENAMEX> </HL>
<DD> <TIMEX TYPE="DATE">06/22/93</TIMEX> </DD>
<SO> WALL STREET JOURNAL (J), PAGE B12 </SO>
<CO>    GTII JNJ </CO>
<IN> BIOTECHNOLOGY (BTC), DRUG MANUFACTURERS (DRG),
   MEDICAL & BIOLOGICAL TECHNOLOGY (MTC) </IN>
<DATELINE> <ENAMEX TYPE="LOCATION">GAITHERSBURG</ENAMEX>, <ENAMEX TYPE="LOCATION">
Md. </ENAMEX> </DATELINE>
<TXT>
<p>
   <ENAMEX TYPE="PERSON">Michael D. Casey</ENAMEX>, a top <ENAMEX TYPE="ORGANIZATION">
Johnson & Johnson</ENAMEX> manager, moved to <ENAMEX TYPE="ORGANIZATION">Genetic
Therapy Inc.</ENAMEX>, a small biotechnology concern here, to become its president
and chief operating officer.
</p>
<p>
   Mr. <ENAMEX TYPE="PERSON">Casey</ENAMEX>, 46 years old, was president of <ENAMEX
TYPE="ORGANIZATION">J&J</ENAMEX>'s <ENAMEX TYPE="ORGANIZATION">McNeil Pharmaceutical
</ENAMEX> subsidiary, which was merged with another <ENAMEX TYPE="ORGANIZATION">J&J
</ENAMEX> unit, <ENAMEX TYPE="ORGANIZATION">Ortho Pharmaceutical Corp.</ENAMEX>,
this year in a cost-cutting move.
</p>
<p>
   Mr. <ENAMEX TYPE="PERSON">Casey</ENAMEX> succeeds <ENAMEX TYPE="PERSON">M. James
Barrett </ENAMEX>, 50, as president of <ENAMEX TYPE="ORGANIZATION">Genetic Therapy
</ENAMEX>. Mr. <ENAMEX TYPE="PERSON">Barrett</ENAMEX> remains chief executive
officer and becomes chairman.
</p>
<p>
   Mr. <ENAMEX TYPE="PERSON">Casey</ENAMEX> said he made the move to the smaller
company because he saw health care moving toward technologies like the company's
gene therapy products. "I believe that the field is emerging and is prepared to
break loose," he said.
</p>
<p>
   Mr. <ENAMEX TYPE="PERSON">Casey</ENAMEX> declined to divulge his compensation,
which he said was comparable to his package at <ENAMEX TYPE="ORGANIZATION">J&J
</ENAMEX>, but acknowledged that the equity portion was higher at <ENAMEX TYPE=
"ORGANIZATION">Genetic Therapy</ENAMEX>.
</p>
<p>
   Noting other recent moves by pharmaceutical executives to small biotech companies,
<ENAMEX TYPE="PERSON">John T.W. Hawkins</ENAMEX>, the executive recruiter who
arranged the <ENAMEX TYPE="ORGANIZATION">Genetic Therapy</ENAMEX> placement, said,
"The equity play is obviously the draw for many of these executives in evaluating
small and emerging companies." But, he added, "fundamentally, it's the excitement
and challenge of building an emerging pharmaceutical company" that attracts the
executives.
```

```
</p>
</TXT>
</DOC>
```

A.3 Coreference Key

```
<DOC>
<DOCID> wsj93_005.0011 </DOCID>
<DOCNO> 930622-0057. </DOCNO>
<HL>    Who's News:
@  <COREF ID="1" MIN="Manager"><COREF ID="3">Johnson & Johnson</COREF>
@  Manager</COREF> Makes <COREF ID="25" MIN="Move">Move
@  To <COREF ID="5">Genetic Therapy</COREF></COREF> </HL>
<DD> 06/22/93 </DD>
<SO> WALL STREET JOURNAL (J), PAGE B12 </SO>
<CO>    GTII JNJ </CO>
<IN> BIOTECHNOLOGY (BTC), DRUG MANUFACTURERS (DRG),
   MEDICAL & BIOLOGICAL TECHNOLOGY (MTC) </IN>
<DATELINE> <COREF ID="7" MIN="GAITHERSBURG">GAITHERSBURG, Md.</COREF> </DATELINE>
<TXT>
<p>
   <COREF ID="0" TYPE="IDENT" REF="1" MIN="Michael D. Casey">Michael D. Casey, a top
<COREF ID="2" TYPE="IDENT" REF="3">Johnson & Johnson</COREF> manager,</COREF> moved
to <COREF ID="4" TYPE="IDENT" REF="5" MIN="Genetic Therapy Inc.">Genetic Therapy
Inc., a small biotechnology concern <COREF ID="6" TYPE="IDENT" REF="7">here</COREF>,
</COREF> to become <COREF ID="9" TYPE="IDENT" REF="0" MIN="president" STATUS="OPT">
<COREF ID="8" TYPE="IDENT" REF="4">its</COREF> president</COREF> and <COREF ID="10"
TYPE="IDENT" REF="0" MIN="officer" STATUS="OPT">chief operating officer</COREF>.
</p>
<p>
   <COREF ID="11" TYPE="IDENT" REF="0" MIN="Casey">Mr. Casey, 46 years old,</COREF>
was <COREF ID="12" TYPE="IDENT" REF="11" MIN="president">president of <COREF ID="13"
TYPE="IDENT" REF="2">J&J</COREF>'s McNeil Pharmaceutical subsidiary, which was merged
with another <COREF ID="14" TYPE="IDENT" REF="13">J&J</COREF> unit, Ortho
Pharmaceutical Corp., this year in a cost-cutting move</COREF>.
</p>
<p>
   <COREF ID="15" TYPE="IDENT" REF="12" MIN="Casey">Mr. Casey</COREF> succeeds
<COREF ID="19" MIN="M. James Barrett">M. James Barrett, 50,</COREF> as <COREF
ID= "16" TYPE= "IDENT" REF="15" MIN="president" STATUS="OPT">president of <COREF
ID="17" TYPE="IDENT" REF="8">Genetic Therapy</COREF></COREF>.
<COREF ID="18" TYPE="IDENT" REF="19" MIN="Barrett">Mr. Barrett</COREF> remains
<COREF ID="20" TYPE="IDENT" REF="18" MIN="officer">chief executive officer</COREF>
and becomes <COREF ID="21" TYPE="IDENT" REF="20">chairman</COREF>.
</p>
<p>
   <COREF ID="22" TYPE="IDENT" REF="15" MIN="Casey">Mr. Casey</COREF> said <COREF
ID="23" TYPE="IDENT" REF="22">he</COREF> made <COREF ID="24" TYPE="IDENT" REF="25"
MIN="move">the move to <COREF ID="26" TYPE="IDENT" REF="17" MIN="company">the
smaller company</COREF></COREF> because <COREF ID="27" TYPE="IDENT" REF="23">he
</COREF> saw health care moving toward technologies like <COREF ID="28" TYPE=
"IDENT" REF="26">the company</COREF>'s gene therapy products. "<COREF ID="29"
TYPE="IDENT" REF="27">I</COREF> believe that the field is emerging and is prepared
to break loose," <COREF ID="30" TYPE="IDENT" REF="29">he</COREF> said.
</p>
<p>
   <COREF ID="31" TYPE="IDENT" REF="30" MIN="Casey">Mr. Casey</COREF> declined to
divulge <COREF ID="32" TYPE="IDENT" REF="31">his</COREF> compensation, which
<COREF ID="33" TYPE="IDENT" REF="32">he</COREF> said was comparable to <COREF
ID="34" TYPE="IDENT" REF="33">his</COREF> package at <COREF ID="35" TYPE="IDENT"
REF="14">J&J</COREF>, but acknowledged that the <COREF ID="41">equity</COREF>
portion was higher at <COREF ID="36" TYPE="IDENT" REF="28">Genetic Therapy</COREF>.
</p>
<p>
   Noting other recent moves by <COREF ID="46" MIN="executives">pharmaceutical
executives</COREF> to small biotech companies, <COREF ID="38" MIN="John T.W. Hawkins">
John T.W. Hawkins, <COREF ID="37" TYPE="IDENT" REF="38" MIN="recruiter">the executive
recruiter who arranged the <COREF ID="39" TYPE="IDENT" REF="36">Genetic Therapy
```

```
</COREF> placement</COREF>,</COREF> said, "<COREF ID="43" MIN="play">The <COREF
ID="40" TYPE="IDENT" REF="41">equity</COREF> play</COREF> is obviously <COREF
ID="42" TYPE="IDENT" REF="43" MIN="draw">the draw for many of these executives in
evaluating small and emerging companies</COREF>." But, <COREF ID="44" TYPE="IDENT"
REF="37">he</COREF> added, "fundamentally, it's the excitement and challenge of
building an emerging pharmaceutical company" that attracts <COREF ID="45" TYPE=
"IDENT" REF="46">the executives</COREF>.
</p>
</TXT>
</DOC>
```

A.4 Coreference Key: Template Version

The start and end offsets are also available, but have been edited out here.

```
<DOCUMENT-9306220057-1> :=
<COREF-9306220057-1> :=
    REF:
    MIN: MANAGER
    TEXT: JOHNSON & JOHNSON @  MANAGER
<COREF-9306220057-3> :=
    REF:
    TEXT: JOHNSON & JOHNSON
<COREF-9306220057-25> :=
    REF:
    MIN: MOVE
    TEXT: MOVE @  TO GENETIC THERAPY
<COREF-9306220057-5> :=
    REF:
    TEXT: GENETIC THERAPY
<COREF-9306220057-7> :=
    REF:
    MIN: GAITHERSBURG
    TEXT: GAITHERSBURG, MD.
<COREF-9306220057-0> :=
    REF: 1
    MIN: MICHAEL D. CASEY
    TEXT: MICHAEL D. CASEY, A TOP JOHNSON & JOHNSON MANAGER,
<COREF-9306220057-2> :=
    REF: 3
    TEXT: JOHNSON & JOHNSON
<COREF-9306220057-4> :=
    REF: 5
    MIN: GENETIC THERAPY INC.
    TEXT: GENETIC THERAPY INC., A SMALL BIOTECHNOLOGY CONCERN HERE,
<COREF-9306220057-6> :=
    REF: 7
    TEXT: HERE
<COREF-9306220057-9> :=
    REF: 0
    STATUS: OPT
    MIN: PRESIDENT
    TEXT: ITS PRESIDENT
<COREF-9306220057-8> :=
    REF: 4
    TEXT: ITS
<COREF-9306220057-10> :=
    REF: 0
    STATUS: OPT
    MIN: OFFICER
    TEXT: CHIEF OPERATING OFFICER
<COREF-9306220057-11> :=
    REF: 0
    MIN: CASEY
    TEXT: MR. CASEY, 46 YEARS OLD,
<COREF-9306220057-12> :=
    REF: 11
```

```
    MIN: PRESIDENT
    TEXT: PRESIDENT OF J&J'S MCNEIL PHARMACEUTICAL SUBSIDIARY, WHICH WAS MERGED
          WITH ANOTHER J&J UNIT, ORTHO PHARMACEUTICAL CORP., THIS YEAR IN A
          COST-CUTTING MOVE
<COREF-9306220057-13> :=
    REF: 2
    TEXT: J&J
<COREF-9306220057-14> :=
    REF: 13
    TEXT: J&J
<COREF-9306220057-15> :=
    REF: 12
    MIN: CASEY
    TEXT: MR. CASEY
<COREF-9306220057-19> :=
    REF:
    MIN: M. JAMES BARRETT
    TEXT: M. JAMES BARRETT, 50,
<COREF-9306220057-16> :=
    REF: 15
    STATUS: OPT
    MIN: PRESIDENT
    TEXT: PRESIDENT OF GENETIC THERAPY
<COREF-9306220057-17> :=
    REF: 8
    TEXT: GENETIC THERAPY
<COREF-9306220057-18> :=
    REF: 19
    MIN: BARRETT
    TEXT: MR. BARRETT
<COREF-9306220057-20> :=
    REF: 18
    MIN: OFFICER
    TEXT: CHIEF EXECUTIVE OFFICER
<COREF-9306220057-21> :=
    REF: 20
    TEXT: CHAIRMAN
<COREF-9306220057-22> :=
    REF: 15
    MIN: CASEY
    TEXT: MR. CASEY
<COREF-9306220057-23> :=
    REF: 22
    TEXT: HE
<COREF-9306220057-24> :=
    REF: 25
    MIN: MOVE
    TEXT: THE MOVE TO THE SMALLER COMPANY
<COREF-9306220057-26> :=
    REF: 17
    MIN: COMPANY
    TEXT: THE SMALLER COMPANY
<COREF-9306220057-27> :=
    REF: 23
    TEXT: HE
<COREF-9306220057-28> :=
    REF: 26
    TEXT: THE COMPANY
<COREF-9306220057-29> :=
    REF: 27
    TEXT: I
<COREF-9306220057-30> :=
    REF: 29
    TEXT: HE
<COREF-9306220057-31> :=
    REF: 30
    MIN: CASEY
    TEXT: MR. CASEY
<COREF-9306220057-32> :=
    REF: 31
    TEXT: HIS
```

```
<COREF-9306220057-33> :=
    REF: 32
    TEXT: HE
<COREF-9306220057-34> :=
    REF: 33
    TEXT: HIS
<COREF-9306220057-35> :=
    REF: 14
    TEXT: J&J
<COREF-9306220057-41> :=
    REF:
    TEXT: EQUITY
<COREF-9306220057-36> :=
    REF: 28
    TEXT: GENETIC THERAPY
<COREF-9306220057-46> :=
    REF:
    MIN: EXECUTIVES
    TEXT: PHARMACEUTICAL EXECUTIVES
<COREF-9306220057-38> :=
    REF:
    MIN: JOHN T.W. HAWKINS
    TEXT: JOHN T.W. HAWKINS, THE EXECUTIVE RECRUITER WHO ARRANGED THE GENETIC
          THERAPY PLACEMENT,
<COREF-9306220057-37> :=
    REF: 38
    MIN: RECRUITER
    TEXT: THE EXECUTIVE RECRUITER WHO ARRANGED THE GENETIC THERAPY PLACEMENT
<COREF-9306220057-39> :=
    REF: 36
    TEXT: GENETIC THERAPY
<COREF-9306220057-43> :=
    REF:
    MIN: PLAY
    TEXT: THE EQUITY PLAY
<COREF-9306220057-40> :=
    REF: 41
    TEXT: EQUITY
<COREF-9306220057-42> :=
    REF: 43
    MIN: DRAW
    TEXT: THE DRAW FOR MANY OF THESE EXECUTIVES IN EVALUATING SMALL AND EMERGING
          COMPANIES
<COREF-9306220057-44> :=
    REF: 37
    TEXT: HE
<COREF-9306220057-45> :=
    REF: 46
    TEXT: THE EXECUTIVES
```

A.5 Template Element Key

```
<ORGANIZATION-9306220057-1> :=
    ORG_NAME: "Johnson & Johnson"
    ORG_ALIAS: "J&J"
    ORG_TYPE: COMPANY
<ORGANIZATION-9306220057-2> :=
    ORG_NAME: "Genetic Therapy Inc."
    ORG_ALIAS: "Genetic Therapy"
    ORG_DESCRIPTOR: "a small biotechnology concern here"
                  / "the smaller company"
    ORG_TYPE: COMPANY
    ORG_LOCALE: GAITHERSBURG CITY
    ORG_COUNTRY: United States
    COMMENT: Locale/Country are from 'here', which refers to dateline
<ORGANIZATION-9306220057-3> :=
    ORG_NAME: "McNeil Pharmaceutical"
    ORG_DESCRIPTOR: "subsidiary"
```

```
    ORG_TYPE: COMPANY
<ORGANIZATION-9306220057-4> :=
    ORG_NAME: "Ortho Pharmaceutical Corp."
    ORG_DESCRIPTOR: "another J&J unit"
    ORG_TYPE: COMPANY
<PERSON-9306220057-1> :=
    PER_NAME: "Michael D. Casey"
    PER_ALIAS: "Casey"
    PER_TITLE: "Mr."
<PERSON-9306220057-2> :=
    PER_NAME: "M. James Barrett"
    PER_ALIAS: "Barrett"
    PER_TITLE: "Mr."
<PERSON-9306220057-3> :=
    PER_NAME: "John T.W. Hawkins"
```

A.6 Scenario Template Key

```
<TEMPLATE-9306220057-1> :=
    DOC_NR: "9306220057"
    CONTENT: <SUCCESSION_EVENT-9306220057-1>
             <SUCCESSION_EVENT-9306220057-2>
             <SUCCESSION_EVENT-9306220057-3>
             <SUCCESSION_EVENT-9306220057-4>
<SUCCESSION_EVENT-9306220057-1> :=
    SUCCESSION_ORG: <ORGANIZATION-9306220057-1>
    POST: "president"
    IN_AND_OUT: <IN_AND_OUT-9306220057-1>
                <IN_AND_OUT-9306220057-2>
    VACANCY_REASON: OTH_UNK
                  / REASSIGNMENT
    COMMENT: "Barrett out, Casey in as pres of Genetic Therapy"
           / "Vacancy at Genetic Therapy either due to predecessor's reassignment
              to chairman or to unknown reasons"
<SUCCESSION_EVENT-9306220057-2> :=
    SUCCESSION_ORG: <ORGANIZATION-9306220057-1>
    POST: "chief operating officer"
    IN_AND_OUT: <IN_AND_OUT-9306220057-3>
    VACANCY_REASON: OTH_UNK
    COMMENT: "Case in as COO of Genetic Therapy"
<SUCCESSION_EVENT-9306220057-3> :=
    SUCCESSION_ORG: <ORGANIZATION-9306220057-2>
    POST: "president"
    IN_AND_OUT: <IN_AND_OUT-9306220057-4>
    VACANCY_REASON: REASSIGNMENT
    COMMENT: "Casey out as pres of McNeil..."
<SUCCESSION_EVENT-9306220057-4> :=
    SUCCESSION_ORG: <ORGANIZATION-9306220057-1>
    POST: "chairman"
    IN_AND_OUT: <IN_AND_OUT-9306220057-5>
    VACANCY_REASON: OTH_UNK
    COMMENT: "Barrett in as chmn of Genetic Therapy"
<IN_AND_OUT-9306220057-1> :=
    IO_PERSON: <PERSON-9306220057-2>
    NEW_STATUS: OUT
    ON_THE_JOB: NO
    OTHER_ORG: <ORGANIZATION-9306220057-1>
    REL_OTHER_ORG: SAME_ORG
    COMMENT: "Barrett out as pres -- stayed with same company"
           / "See IN_AND_OUT-2 re reasoning for ON_THE_JOB fill"
<IN_AND_OUT-9306220057-2> :=
    IO_PERSON: <PERSON-9306220057-1>
    NEW_STATUS: IN
    ON_THE_JOB: YES
    OTHER_ORG: <ORGANIZATION-9306220057-2>
    REL_OTHER_ORG: OUTSIDE_ORG
    COMMENT: "Casey in as pres -- came from different org (see separate event)"
           / "He's probably on the job already (because he has already 'made the
```

```
            move'), but it's not entirely clear"
<IN_AND_OUT-9306220057-3> :=
    IO_PERSON: <PERSON-9306220057-1>
    NEW_STATUS: IN
    ON_THE_JOB: YES
    OTHER_ORG: <ORGANIZATION-9306220057-2>
    REL_OTHER_ORG: OUTSIDE_ORG
    COMMENT: "Casey in as COO"
           / "See IN_AND_OUT-2 re fill for ON_THE_JOB"
<IN_AND_OUT-9306220057-4> :=
    IO_PERSON: <PERSON-9306220057-1>
    NEW_STATUS: OUT
    ON_THE_JOB: NO
    OTHER_ORG: <ORGANIZATION-9306220057-1>
    REL_OTHER_ORG: OUTSIDE_ORG
    COMMENT: "Casey out as pres of McNeil... -- went to diff org (see separate event)"
           / "It's clear he's not on the job at McNeil any more; it's just not
              totally clear that he is on the job at Genetic Therapy yet"
<IN_AND_OUT-9306220057-5> :=
    IO_PERSON: <PERSON-9306220057-2>
    NEW_STATUS: IN
    ON_THE_JOB: UNCLEAR
    OTHER_ORG: <ORGANIZATION-9306220057-1>
    REL_OTHER_ORG: SAME_ORG
    COMMENT: "Barrett in -- acquiring new title at same org"
           / "ON_THE_JOB: 'becomes chairman'"
<ORGANIZATION-9306220057-1> :=
    ORG_NAME: "Genetic Therapy Inc."
    ORG_ALIAS: "Genetic Therapy"
    ORG_DESCRIPTOR: "a small biotechnology concern here"
                  / "a small biotechnology concern"
                  / "the smaller company"
    ORG_TYPE: COMPANY
    ORG_LOCALE: GAITHERSBURG CITY
    ORG_COUNTRY: United States
    COMMENT: Locale/Country are from 'here', which refers to dateline
<ORGANIZATION-9306220057-2> :=
    ORG_NAME: "McNeil Pharmaceutical"
    ORG_TYPE: COMPANY
<PERSON-9306220057-1> :=
    PER_NAME: "Michael D. Casey"
    PER_ALIAS: "Casey"
    PER_TITLE: "Mr."
<PERSON-9306220057-2> :=
    PER_NAME: "M. James Barrett"
    PER_ALIAS: "Barrett"
    PER_TITLE: "Mr."
```

Appendix B

The Template Comparison Tool

The basic purpose of this tool is to compare any number of responses in template form to a key, and to output the comparison in a concise and useful form.

It works directly on TE and ST templates, and with a bit more work, with NE and CO when they are converted to template form[1]. The comparison tool makes use of the map histories produced during scoring, thus can be used to check scorer functionality as well as system performance[2]. Several extensive examples of the tool's use are in the thesis chapter on LOLITA performance, in particular section 4.4 which contains an example of multi-system comparison for TE, and appendix C which contains a worked example of Correctness Analysis (see section 5.6). This appendix also contains a short example (see section B.2).

The tool is written in `perl`[3], and is around 2000 lines long. It uses a freely available library of Set operations. The author intends to make this tool and related scripts publicly available. Note that it is essentially a prototype written to help explore the MUC-6 output, and not production-quality software: several weaknesses exist, with an approximate error rate of 3-4% (see section B.6). Please contact the author for more details (`P.C.Callaghan@uk.ac.durham`).

B.1 Explanation of Symbols

The output follows the structure of the key templates, inserting extra information to show how individual systems performed. The following symbols are used in the output. Please refer to examples to help understand how the symbols are used.

In the examples of each (following `# eg`), the left side of the `<->` is the key, the right side a system's response. `A, B, C` represent slot fills, and `A B` represents a multiple fill.

```
$exact_equal   = "==";  # as it says
                        # eg A B C <-> A B C
$wrong         = "XX";  # no match
                        # eg A     <-> C
$omitted       = "--";  # system does not produce anything
                        # eg A     <-> (?)
$inserted      = "++";  # no correspondence with answer
                        # eg (?)   <-> A
$partial       = ".5";  # an intersection with the answer & not subset
                        # eg A B   <-> A C
$too_little    = "<<";  # non-empty proper subset of an answer
                        # eg A B C <-> A B
```

[1] This is currently done with version 3.2 of the MUC scorer by post-processing the IE format of the report summary files.

[2] These are also extracted from the IE format of the the report summary files.

[3] See, for example, `http://www.perl.com/perl/index.html`

```
$too_much      = ">>";  # superset of a non-empty answer
                        # eg A B   <-> A B C
$no_key_answer = "??";  # = don't know, is for unmatched templates
$wrong_slot    = "##";  # used for slot without complete match.
```

Template names in angle brackets (<...>) indicate a key template, in braces ({...}) a response template. To avoid confusion, response template names are prefixed with the identifier for the system, eg a-{...}. In cases of alternative fills, if all systems scoring on the whole template produce the same alternative, it is marked with a '*'.

B.2 An Example

This example template is for illustration only - it is not a proper MUC-6 template. Thus, the contents are not intended to be consistent and correct.

```
<IN_AND_OUT-5> :=
        a    {IN_AND_OUT-966820096754}       sc 75.00
        b    {IN_AND_OUT-1}                  sc 60.00
        g    {IN_AND_OUT-3}                  sc 60.00
        j    {IN_AND_OUT-1}                  sc 60.00
        k    {MISMATCHED OR UNDERGENERATED}  {ignoring in comparison}
```

Angle-brackets contain a key template name. The next five lines indicate matches to this template, with the (f-) scores awarded. The names in braces are a system's own name for the template mapped. Systems are identified by lower case letters.

The final line indicates that 'k' did not produce a template which was matched by the scorer. 'k' will NOT be mentioned in the remainder of the template - else it would produce many "missing" slots.

```
## REL_OTHER_ORG:    SAME_ORG
              ==     c d e f i j
              XX  b= OUTSIDE_ORG
              XX  g= OUTSIDE_ORG
              --     a h
              XX     SOMETHING_ELSE
                  XX l m n o p
```

This is a value slot - REL_OTHER_ORG. Something is wrong, so it is prefixed with ##. Systems {c d e f i j} produced an exact match to the key answer, "SAME_ORG" whereas {b g} said "OUTSIDE_ORG" and {a h} produced nothing. When more than two systems have the same wrong answer, the display will be condensed as shown in the last two lines. (NB SOMETHING_ELSE is an invented possible fill for the purposes of this example only.)

```
>    NEW_STATUS:       IN
```

Everyone (except k, of course) produced the correct answer.

```
## ORG_DESCRIPTOR:
                     "a small biotechnology concern here"
                   / "a small biotechnology concern"
              ==     b c g
                   / "the smaller company"
              XX  h= "the company"
              XX  j= "J&J's McNeil Pharmaceutical subsidiary"
              XX  d= "pharmaceutical company"
              XX  e= "pharmaceutical company"
              --     a i
```

An example of a multi-valued slot. {b c g} matched one alternative (the second). {h j d e} were all wrong, and {a i} did not produce an answer. Where everyone is correct, unanimous agreement on one alternative is marked with an asterisk. If a system produces more than one fill, the fills are separated by a colon.

A further comment on format: the task specifications do not specify whether multi-word answers should be quoted. Quotes are required to disambiguate multiple fills which are themselves multi-word phrases (eg "the small bank" and "the bank where I work"). The script sometimes adds quotes to strings when 'tokenising' the slot fills to make the string formats consistent before the set comparison operations, and removes all quotes from the comparison output. Multiple fills are separated by colons, which delimits multi-word strings equally as well as quotes. This strategy does not cause matches to be rejected, just that quotes in the key or responses are not shown. It would be useful if the task specifications required that *all* strings from an article be quoted: for example, the 'grammar' for templates could include a specification of legal fills[4]. This requirement would be similar to the current requirement that all template pointers be enclosed in angle brackets (which is not strictly necessary as the type can be inferred from the specification of the slot in the configuration files).

```
## SOME_SLOT:          1 2
               ==       b
               >>   c= 1 2 3
               <<   a= 1
               .5   d= 1 3
               .5   e= 1 3
               .5   f= 1 3 4
                     / 3 4
               >>   f= 1 3 4
               .5   c= 1 2 3
               .5   d= 1 3
               .5   e= 1 3
               --       g
```

This is more thorough example of the alternative set handling, showing too much (>>), too little (<<), and an intersection (.5) of the required answers.

```
OTHER_ORG:
     <ORGANIZATION-1>     --    a {ORGANIZATION-96580}       sc 66.67
                               b {ORGANIZATION-1}           sc 80.00
                               c {ORGANIZATION-1}           sc 80.00
                               d {ORGANIZATION-1}           sc 60.00
                               e {ORGANIZATION-1}           sc 60.00
                          --   f {no response equiv}
                               g {ORGANIZATION-20}          sc 80.00
                          --   h {ORGANIZATION-113}         sc 60.00
                               i {ORGANIZATION-2}           sc 66.67
                               j {ORGANIZATION-3}           sc 60.00
     <no key equiv>       ++   f {ORGANIZATION-1}
```

A link slot, with one link. {b c d e g i j} were all correct. The scores for the corresponding template are shown on the right. {a f h} didn't produce this link; {a h}'s equivalent of the expected template are shown, but {f} had no equivalent. Finally, {f} thought its ORGANIZATION-1 should be here (and this was not matched to a key template). This under-generating and over-generating could be a case of system f producing the wrong link, but this is not deducible from the results, hence the output does not reflect this possibility. Link slots containing errors are *not* marked with ## as are value slots.

[4]The tool does not currently make use of the scorer slot configuration files, which may help with the problems mentioned.

B.3 Coref mode

Coref templates, as extracted from the scorer's IE format report summary files, contain only one slot amenable to direct comparison, the `TEXT` slot. Thus, the normal template content comparison is not that useful, since it is the relationships between chains of templates (ie, markups) which are important.

The template mapping information is also useful: it allows representation of the matches made by the scorer. Unfortunately, the f-scores of normal templates are not well-defined for coref, and no other simple scores are produced in their place. The author is currently experimenting with ways of presenting information produced during scoring (see section 4.3).

The tool adds a slot `KEY_CHAIN` to all templates, to indicate which chain a template belongs to. For key templates, the value of `KEY_CHAIN` is identical for all members of a chain. For response templates that are mapped to a key template, this is the `KEY_CHAIN` of the key template. For unmapped response templates, this slot is not filled. Response templates also get a similar slot `OWN_CHAIN` which identifies the chain in a system's response. Note that these two slots replace the ID and REF system of denoting chains.

B.4 Graph Output

Several graphical representations of ST and CO input can be produced. Currently, graphs for the daVinci visualisation tool [Fröhlich and Werner, 1997] are output, although conversion to other graph tools is possible.

One representation is of the relationship between ST templates in the key or response. Additionally, nodes in the key version can be shown with the content of the response node which the key node is mapped to. An example appears in figure 4.3.

For Coref, we can graph the correspondance between chains in the key and response, as in figure 4.1. The heart of this diagram is the linkage between markups in the key to those in the response. Markups are then grouped into key chains (above) and response chains (below). When the intersections between chains is complex, a second mode is possible, which groups dependent chains together, thus separating them from 'normal' chains. An example of intersection is where some response chains `L_0` and `L_1` both map in to key chains `K_2` and `K_3`.

B.5 Statistics

The tool can collect statistics, as described in section 5.6 and section 5.7 of the thesis. These are not guaranteed absolutely correct, mainly for the reasons outlined in the next section. When these points are not involved, the tool's collection of statistics agrees very well with the conventional scorers, so we claim that basically the tool is quite accurate.

B.6 Limitations

The following points contribute to deviations from the behaviour of the conventional scorers. One important point is that some of these limitations are due to implementation choices. To remove them would require a much more complicated implementation. For the present exploratory work, this effort would not be justified by the small improvement possible (a few percent).

A further point is that the MUC scorer has been much improved since the tool was written. It is now feasible to reproduce the functionality of the tool by post-processing the report summary files of the scorer because the report summary files contain all the basic information required. Hence new implementations of the comparison tool should be based on such files.

- We do not implement the minimum string convention for coref matches, although this

information is implied by the inferred KEY_CHAIN slot being filled in a response template: a match has been given.

- The string matching is not as sophisticated as the official scorer's. It is correct for almost all cases, the exceptions being matches involving disjoint sets of multiple fills, and strings containing the rare corporate designators (only the important ones have been implemented).

- All optional material is considered as scorable, and hence processed as if it was not optional. The main reason for this is that correct implementation is complicated and a reimplementation would be required (ie, it is not a trivial change). For example, each system produces different amounts of the optional material. Optional material could obviously be ignored, but it does add to task performance, so we feel bound to handle it in some way.

  For informativeness, the tool shows the performance of each system on every part of the key, so users could ignore the optional parts if desired. The question of optionality is more serious for the statistical analyses. The author does not see a natural way to treat optional material with correctness analysis. It is surely significant if no system attempted the optional material. Under this view, it does not seem unreasonable to compare systems on all possible material, optional or not. Note that all systems are thus compared on the same material.

  A special case of "empty alternatives" has been implemented, for values where absence of fill is acceptable. When a system does not produce a value for slot X, and the slot allows the empty alternative, then the system is treated as if it was correct for that slot.

  Optional material becomes more significant in ST, where optional links make templates and their children optional by inference. It appears that around 10% of the scorable features in the ST key are optional in some way (this is suggested by the difference between POS values in the scorer results and the count of scorable material produced by this tool). No special treatment of ST material has been implemented, apart from the case of empty alternatives, and alternative link sets are replaced by a single link set which is the union of the alternatives (this heuristic is reasonable, since many uses of alternative link sets is to make one link optional, eg `A B C / A B C D` to make `D` optional).

- The arbitrary pairing of unmatched templates (within type) to produce simple incorrectness errors instead of overgeneration and undergeneration is not reproduced. Since this is arbitrary, we cannot be sure of reproducing it. Furthermore, the author does not agree with this policy.

- Alternative multiple links are incorrectly handled in the template tasks keys. This also affects counting of the `<<`, but such slots only occur a few times in a thousand slots.

- Lastly, there are probably some small bugs remaining.

Appendix C

Worked Example of Correctness Analysis

C.1 Introduction

This appendix contains a detailed worked example of performing correctness analysis for three systems on a simple article.

What we describe here is the basic method, sufficient to process the example TE-based results given. We omit specific details on harder points, eg of how to handle links, alternatives, and multiple fills. See appendix B for such details.

During production of the template comparison output for a single article, data is collected (see section C.3), producing tables of the basic information for that article (see section C.4). This information is then collated across articles (see section C.5). To explain how the data is collected, we begin with the details of an abstract machine.

C.2 An Abstract Machine for Collecting Score Information

The data collection is implemented as operations on an abstract machine, with the following basic interface. Operations will be done on the abstract machine as an article is analysed (see section C.3), and hence the collection process may be understood by seeing which operations are called for which input. There are some other less important commands, eg related to matching just one alternative from a multiple fill; we do not discuss them here. Note that this interface is only used for slot instantiations appearing in the key, since correctness analysis is only defined for this common material.

- `begin_slot SLOT_NAME`: This command marks the start of a new slot instantiation of given type. Each slot instantiation is treated separately. Information from the other commands is accumulated until `end_slot` is received. Several calls of each command are allowed, eg calling `correct` for each system that was judged correct.

- `end_slot`: The information collected since the `begin_slot` is added to internal tables. These tables are indexed by correctness class and name of slot, and represent the state of knowledge over all slots (in all templates) thus far processed. The correctness class is determined by the number of systems declared correct since the `begin_slot` call. Firstly, we store the number of occurrences of the relevant combination of correctness class and slot. Then, two values are stored for each system: how many times it is correct in the combination, and the same for the number incorrect. Additionally for each system we store the global counts of over-generation and under-generation in order to assess accuracy (checking against scorer output). The tables are described in section C.4.

- `correct SYSTEM1, SYSTEM2, ...`: The listed systems were correct for this slot.
- `incorrect SYSTEM1, SYSTEM2, ...`: The listed systems produced in incorrect fill.
- `undergen SYSTEM1, SYSTEM2, ...`: The listed systems did not produce a fill. This includes omission of slot instantiation in a template, and omission of instantiations because the relevant template was not produced.

A separate command `overgen SYSTEM1, SYSTEM2, ...` is used to register a system overgenerating at any point. It can be called at any time – inside a proper template where the system has produced a slot instantiation not appearing in the key, or for all parts of response templates which are not mapped to anything in the key.

C.3 Worked Example: Data Collection

A section of the TE performance of three systems on a simple article is represented below as output of the comparison tool. It was based on article 9404130062, but some templates have been removed, to keep the example short, and to add some more errors, a new person template was added and some slot fills made incorrect. For each template we list the calls to the abstract machine.

- <ORGANIZATION-1> :=

```
        b    {ORGANIZATION-2}                sc 80.00
        c    {ORGANIZATION-3}                sc 72.73
        a    {MISMATCHED OR UNDERGENERATED}  {ignoring in comparison}
## ORG_ALIAS:          Burns Fry
                ==      b
                XX  c= Burns Fry Something
## ORG_COUNTRY:        Canada
                ==      c
                --      b
## ORG_DESCRIPTOR:     this brokerage firm
                ==      b
                --      c
## ORG_LOCALE:         Toronto CITY
                ==      c
                --      b
   ORG_NAME:          Burns Fry Ltd
   ORG_TYPE:          COMPANY

begin_slot ORG_ALIAS
correct b
incorrect c
undergen a
end_slot
begin_slot ORG_COUNTRY
correct c
undergen b, a
end_slot
begin_slot ORG_DESCRIPTOR
correct b
undergen c, a
end_slot
begin_slot ORG_LOCALE
correct c
undergen b, a
end_slot
begin_slot ORG_NAME
correctb, c
undergen a
end_slot
begin_slot ORG_TYPE
correct b, c
undergen a
end_slot
```

```
• <PERSON-1> :=
        a    {PERSON-96300}                  sc 100.00
        b    {PERSON-2}                      sc 100.00
        c    {PERSON-2}                      sc 100.00
   PER_ALIAS:          Wright
   PER_NAME:           Donald Wright
   PER_TITLE:          Mr

  begin_slot PER_ALIAS
  correct a, b, c
  end_slot
  begin_slot PER_NAME
  correct a, b, c
  end_slot
  begin_slot PER_TITLE
  correct a, b, c
  end_slot

• <ORGANIZATION-2> :=
        a    {ORGANIZATION-96302}            sc 100.00
        b    {ORGANIZATION-1}                sc 85.71
        c    {ORGANIZATION-1}                sc 80.00
  ## ORG_ALIAS:          Merrill Lynch
                  ==     b
                  --     a c
  ## ORG_DESCRIPTOR:     a unit of Merrill Lynch & Co
                  ==     a
                  --     b c
   ORG_NAME:           Merrill Lynch Canada Inc
   ORG_TYPE:           COMPANY

  begin_slot ORG_ALIAS
  correct b
  undergen a,c
  end_slot
  begin_slot ORG_DESCRIPTOR
  correct a
  undergen b, c
  end_slot
  begin_slot ORG_NAME
  correct a, b, c
  end_slot
  begin_slot ORG_TYPE
  correct a, b, c
  end_slot

• <PERSON-3> :=
        a    {MISMATCHED OR UNDERGENERATED}   {ignoring in comparison}
        b    {MISMATCHED OR UNDERGENERATED}   {ignoring in comparison}
        c    {MISMATCHED OR UNDERGENERATED}   {ignoring in comparison}
   PER_NAME:           Fred Smith

  begin_slot PER_NAME
  undergen a, b, c
  end_slot

• <MISMATCHED or OVERGENERATED> := a-{PERSON-96295}
  ?? PER_NAME:          FRY Ltd

  overgen a
```

C.4 Worked Example: Tables

The result after data collection is a table, whose basic form was explained on p. 187. Adopting MUC6 terminology, we thus have for a correctness class C and a slot T, we have POS, the number of times the combination of C and T occurred, plus for a system S, two values COR and INC. Since overgeneration is not considered in correctness analysis, we can calculate the ACT value for a system (ie, the number of attempts) as just the sum of COR and INC.

So, for the combination of C and T, we can calculate recall REC for S as COR / POS, and precision PRE as COR / ACT. We can then calculate the f-score as defined in MUC-6, ie $((1 + \beta^2)$ * REC * PRE) / (β^2 * PRE + REC). (We take β^2 as 1.0, as standard.) Note that these are undefined if the denominator is zero.

Several views of this information are possible. Most useful is where we discard the slot name information and consider just correctness classes. Such tables for each system are given in the second subsection.

Note that the template comparison tool and the implementation of data collection agrees perfectly with the official MUC scorer for this example.

C.4.1 Basic Table

This is shown with lines of the following format. The lines are grouped by slot and correctness class.

```
<T>_<C>  (<POS>) =  <COR> y, <INC> n, <REC> <PRE> <F-score> :: <WHO>
```

The tables are compressed when systems have identical information; so `WHO` represents the systems who got the given `COR` and `INC` combination. The remaining variables were introduced above.

PERSON Slots

```
 PER_NAME_0 (1) = 0 y,  0 n, 0.0000     --     -- r-p-f :: abc

 PER_NAME_3 (1) = 1 y,  0 n, 1.0000 1.0000 1.0000 r-p-f :: abc

PER_ALIAS_3 (1) = 1 y,  0 n, 1.0000 1.0000 1.0000 r-p-f :: abc

PER_TITLE_3 (1) = 1 y,  0 n, 1.0000 1.0000 1.0000 r-p-f :: abc
```

A few words of explanation. There was one PER_NAME where zero systems were correct; in this case, all systems under-generated, hence a zero INC value. There was one PER_NAME where all three systems were correct; obviously they each get one for COR and zero for INC, with corresponding REC values etc. This is true for the other PER_ slots.

ORGANIZATION Slots

```
 ORG_NAME_2 (1) = 0 y,  0 n, 0.0000     --     -- r-p-f :: a
 ORG_NAME_2 (1) = 1 y,  0 n, 1.0000 1.0000 1.0000 r-p-f :: bc

 ORG_NAME_3 (1) = 1 y,  0 n, 1.0000 1.0000 1.0000 r-p-f :: abc

 ORG_TYPE_2 (1) = 0 y,  0 n, 0.0000     --     -- r-p-f :: a
 ORG_TYPE_2 (1) = 1 y,  0 n, 1.0000 1.0000 1.0000 r-p-f :: bc

 ORG_TYPE_3 (1) = 1 y,  0 n, 1.0000 1.0000 1.0000 r-p-f :: abc

ORG_ALIAS_1 (2) = 0 y,  0 n, 0.0000     --     -- r-p-f :: a
ORG_ALIAS_1 (2) = 0 y,  1 n, 0.0000 0.0000     -- r-p-f :: c
ORG_ALIAS_1 (2) = 2 y,  0 n, 1.0000 1.0000 1.0000 r-p-f :: b

ORG_LOCALE_1 (1) = 0 y,  0 n, 0.0000     --     -- r-p-f :: ab
ORG_LOCALE_1 (1) = 1 y,  0 n, 1.0000 1.0000 1.0000 r-p-f :: c

ORG_COUNTRY_1 (1) = 0 y,  0 n, 0.0000     --     -- r-p-f :: ab
ORG_COUNTRY_1 (1) = 1 y,  0 n, 1.0000 1.0000 1.0000 r-p-f :: c
```

```
ORG_DESCRIPTOR_1 (2) = 0 y,  0 n, 0.0000     --     -- r-p-f :: c
ORG_DESCRIPTOR_1 (2) = 1 y,  0 n, 0.5000 1.0000 0.6667 r-p-f :: ab
```

For ORG_NAME_2, where two systems were correct on one occasion, we have two lines. The first says system 'a' did not produce a fill for this (since it was not correct, and not incorrect). The second says 'b' and 'c' both were correct.

ORG_ALIAS is the most interesting case. There were two occurrences of just one system being correct for an ORG_ALIAS instantiation. System 'a' was neither correct not incorrect in both (ie, under-generated). System 'c' was incorrect in one of these occurrences (and under-generated in the other). System 'b' was correct in both cases.

C.4.2 Tables Without Slot Information

To reduce the information, we can discard the slot information, so the information is indexed by correctness class alone. To obtain COR, INC, ACT values, we sum these values over the slots and recalculate REC, PRE, etc. We can also present the information specific to a system, hence obtain the tables below. To help assess accuracy, a final line summarises each table, listing POS, ACT, COR, INC, OVG, REC, PRE, and F-score in that order.

- TABLE for system a:

```
  a     0(1)=   0   0, 0.0000     --     -- r-p-f
  a     1(6)=   1   0, 0.1667 1.0000 0.2858 r-p-f
  a     2(2)=   0   0, 0.0000     --     -- r-p-f
  a     3(5)=   5   0, 1.0000 1.0000 1.0000 r-p-f
Summary a = 14 c, 7 a, 6 y, 0 n, 1 o :: 0.4286 0.8571 0.5714 r-p-f
```

  We can see that 'a' was the system correct on one of the six occasions where only one was correct. Obviously it was correct for all five of the occasions when all three were correct. There was one occasion where no-one was correct, and 'a' under-generated here (ie, it did not produce an incorrect fill). It also under-generated on the two occasions when two systems were correct. Finally, 'a' overgenerated once (represented by `1 o` in the summary line).

- TABLE for system b:

```
  b     0(1)=   0   0, 0.0000     --     -- r-p-f
  b     1(6)=   3   0, 0.5000 1.0000 0.6667 r-p-f
  b     2(2)=   2   0, 1.0000 1.0000 1.0000 r-p-f
  b     3(5)=   5   0, 1.0000 1.0000 1.0000 r-p-f
Summary b = 14 c, 10 a, 10 y, 0 n, 0 o :: 0.7143 1.0000 0.8333 r-p-f
```

  System 'b' under-generated where no-one was correct, was the only system correct on three occasions of the possible six, was one of the two correct on both occasions, and obviously was correct whenever everyone was correct. It was not incorrect anywhere, and did not over-generate.

- TABLE for system c:

```
  c     0(1)=   0   0, 0.0000     --     -- r-p-f
  c     1(6)=   2   1, 0.3333 0.6667 0.4444 r-p-f
  c     2(2)=   2   0, 1.0000 1.0000 1.0000 r-p-f
  c     3(5)=   5   0, 1.0000 1.0000 1.0000 r-p-f
Summary c = 14 c, 10 a, 9 y, 1 n, 0 o :: 0.6429 0.9000 0.7500 r-p-f
```

  It was the only system correct on two occasions of the six, and was incorrect for one of the other occasions. Otherwise it follows the pattern of 'c'.

The last table summarises the information over all systems. It tabulates for each correctness class the size of the class (ie, its POS value), plus the F-scores in that class of each system. Furthermore, each F-score is multiplied by the POS of the relevant class in order to be able to show both F-score and class size on the same graph. It also gives an idea of how the F-scores contribute to the overall score (ie a high F-score in a class of size 10 is worth less than a not so high F-score in a class of size several hundred). Undefined values, principally where the denominator in some calculation was zero, are shown as `--`.

```
TABLE: Count-scaled F-score table:
  0    1 =       --       --       --
  1    6 =   1.7148   4.0002   2.6664
  2    2 =       --        2        2
  3    5 =        5        5        5
```

C.5 Combining Across Articles

Combining the tables across articles works as if the articles were analysed in sequence: thus for each correctness class and slot combination, the POS counts are summed, and for each system the COR and INC counts are summed. The independent over-generation count for each system is also summed. Then, system-specific tables can be prepared from the main table and the derived statistics recalculated. For example, combining the article above with itself, we obtain for the final table as below. This can now be displayed as a graph.

```
TABLE: Count-scaled F-score table:
  0    2 =       --       --       --
  1   12 =   3.4296   8.0004   5.3328
  2    4 =       --        4        4
  3   10 =       10       10       10
```

Appendix D

Size Ranks of MUC-6 Articles

id	rank	id	rank	id	rank	id	rank
9307260024	1	9402160106	2	9401120067	3	9305170164	4
9309140164	5	9307130174	6	9402220071	7	9312230003	8
9308230127	9	9305120155	10	9404130168	11	9307080118	12
9404130062	13	9303310131	14	9312030175	15	9404080155	16
9305050122	17	9304010017	18	9310040154	19	9306100111	20
9306240111	21	9402100124	22	9303110125	23	9401040159	24
9305070042	25	9304020097	26	9401130019	27	9302030136	28
9301130133	29	9401030048	30	9401040117	31	9301190125	32
9404150071	33	9404270051	34	9404200037	35	9306070139	36
9401110053	37	9309100115	38	9403090076	39	9401050050	40
9311150068	41	9403110035	42	9303020074	43	9307190045	44
9310280136	45	9304190138	46	9401250091	47	9403230168	48
9306220057	ch	9404250056	50	9306040089	51	9309100116	52
9401270106	53	9403040124	54	9401210129	55	9309230076	56
9402030012	57	9402150012	58	9404070005	59	9403080001	60
9310040005	61	9404070015	62	9403290146	63	9401190015	64
9310190008	65	9401200153	66	9402110060	67	9401270084	68
9303250020	69	9403100063	70	9402180145	71	9404080111	72
9401130054	73	9401100060	74	9306210166	75	9404110093	76
9402230039	77	9302100071	78	9310080022	79	9403160006	80
9311020154	81	9301190098	82	9301060123	83	9402240049	84
9307290143	85	9306280018	86	9403230090	87	9403140041	88
9404140120	89	9307220047	90	9402240133	wt	9303190092	92
9402180067	93	9404010088	94	9403180009	95	9404040040	96
9305040023	97	9308110045	98	9401270105	99	9308200068	100

References

[Allen, 1995] J. Allen, *Natural Language Understanding, 2nd Edn*, AW, 1995.

[Alshawi *et al.*, 1992] H. Alshawi, D. Carter, R. Crouch, S. Pulman, M. Rayner, and A. Smith, "CLARE: A Contextual Reasoning and Cooperative Response Framework for the Core Language Engine (Final Report)", Technical report, SRI Cambridge, Dec 1992.

[Augusstson, 1996] L. Augusstson, "HBC - The Chalmers Haskell compiler", `http://-www.cs.chalmers.se/~augustss/hbc.html, 1996`.

[Balkan *et al.*, 1995] L. Balkan, D. Arnold, and F. Fouvry, "Test Suites for Evaluation in Natural Language Engineering", in *Proc 2nd Language Engineering Conf*, 1995.

[Baring-Gould, 1997] S. Baring-Gould, *SemNet: the Knowledge Representation of LOLITA*, PhD thesis, Department of Computer Science, University of Durham, 1997, (submission due in 1997).

[Black *et al.*, 1991] E. Black, S. Abney, D. Flickenger, C. Gdaniec, R. Grishman, P. Harrison, D. Hindle, R. Ingria, F. Jelinek, J. Klavans, M. Liberman, M. Marcus, S. Roukos, B. Santorini, and T. Strzalkowski., "A procedure for quantitatively comparing the syntactic coverage of English grammars", in *Proceedings of the Fourth DARPA Speech and Natural Language Workshop*, Feb 1991.

[Black, 1996] E. Black, "Evaluation of Broad-Coverage Natural-Language Parsers", `http://-www.cse.ogi.edu/CSLU/HLTsurvey/ch13node6.html, 1996`.

[Boguraev *et al.*, 1995] B. Boguraev, R. Garigliano, and J. Tait, "Editorial", *Journal of Natural Language Engineering*, 1(1), 1995.

[Brill, 1995] E. Brill, "Rule-based tagger", email:brill@cs.jhu.edu, 1995.

[Callaghan *et al.*, 1994] P. C. Callaghan, R. G. Morgan, and R. Garigliano, "Prosodic Assignment in a Large-scale Natural Language Processing System", in *Proc. Institute of Acoustics*, page ., Nov 1994.

[Carlsson and Hallgren, 1993] M. Carlsson and T. Hallgren, "FUDGETS: A Graphical User Interface in a Lazy Functional Language", in *Proceedings of Conference on Functional Programming Languages and Computer Architecture (FPCA 93)*, pages 321–330, June 1993.

[Chinchor and Dungca, 1995] N. Chinchor and G. Dungca, "Four scorers and seven years ago: the scoring method for MUC-6", in *Proceedings: Sixth Message Understanding Conference (MUC-6)*, pages 33–38, November 1995.

[Chinchor, 1993] N. Chinchor, "The statistical significance of evaluation results", Technical Report 2559, SAIC, October 1993.

[Chinchor, 1995a] N. Chinchor, "MUC-6 Scoring System User's Manual", distributed with scoring software, Jan 1995.

[Chinchor, 1995b] N. Chinchor, "Statistical Significance of MUC-6 Results", in *Proceedings: Sixth Message Understanding Conference (MUC-6)*, pages 39–43, November 1995.

[Cole *et al.*, 1996] R. A. Cole, J. Mariani, H. Uszkoreit, A. Zaenen, and V. Zue, *Survey of the State of the Art in Human Language Technology*, `http://www.cse.ogi.edu/CSLU/HLTsurvey/-HLTsurvey.html, 1996`.

[Collingham, 1995] R. J. Collingham, *An Automatic Speech Recognition System for use by Deaf Students in Lectures*, PhD thesis, Department of Computer Science, University of Durham, 1995.

[CRL, 1997] CRL, "Tabula Rasa", `http://crl.nmsu.edu/Research/Projects/tr, 1997`, Computing Research Laboratory, New Mexico State University.

[Crouch *et al.*, 1995] R. Crouch, R. Gaizauskas, and K. Netter, "Report of the Study Group on Assessment and Evaluation", `http://xxx.lanl.gov/ps/cmp-lg/9601003, April 1995`.

[Cunningham *et al.*, 1995] H. Cunningham, R. Gaizauskas, and Y. Wilks, "A General Architecture for Text Engineering (GATE) - a new approach to Language Engineering R&D", Technical Report 21, University of Sheffield, 1995.

[DARPA, 1995] DARPA, *Proceedings of the Sixth Message Understanding Conference (MUC-6)*, Morgan Kaufman, 1995.

[EAGLES, 1994] EAGLES, "Evaluation of Natural Language Processing Systems - INTERRIM REPORT", Technical report, EAGLES, Oct 1994, `http://issco-www.unige.ch/ewg95/-ewg95.html`.

[EAGLES, 1995] EAGLES, "Evaluation of Natural Language Processing Systems - FINAL REPORT", Technical report, EAGLES, Sept 1995, `http://issco-www.unige.ch/ewg95/-ewg95.html`.

[EAGLES, 1997] EAGLES, "EAGLES WWW Page", `http://www.ilc.pi.cnr.it/EAGLES/-home.html, 1997`.

[Ellis *et al.*, 1993] N. Ellis, R. Garigliano, and R. Morgan, "A New Transformation into Deterministically Parsable Form for Natural Language Grammars", in *Proceedings of Sigparse*, 1993.

[Fernández, 1995] M. Fernández, "Spanish Generation in the NL System LOLITA", Master's thesis, Dept of Computer Science, Univeristy of Durham, 1995.

[FraCaS, 1995] FraCaS, "Evaluating the State of the Art (deliverable 10)", `http://-www.cogsci.ed.ac.uk/~fracas/, Jan 1995`.

[Fröhlich and Werner, 1997] M. Fröhlich and M. Werner, "The daVinci Graph Visualisation Tool", `http://www.informatik.uni-bremen.de:80/~inform/forschung/daVinci`, 1997.

[Frost and Launchbury, 1989] R. Frost and J. Launchbury, "Constructing Natural Language Interpreters in a Lazy Functional Language", *The Computer Journal*, 32(2):108–121, 1989.

[Galliers and Sparck Jones, 1993] J. Galliers and K. Sparck Jones, "Evaluating Natural Language Processing Systems", Technical Report 291, University of Cambridge Computer Laboratory, 1993, (Note: now published as [Sparck Jones and Galliers, 1996].).

[Garigliano *et al.*, 1993] R. Garigliano, R. G. Morgan, and M. H. Smith, "The LOLITA System as a Contents Scanning Tool", in *Proceedings of the 13th International Conference on Artificial Intelligence, Expert Systems and Natural Language Processing*, May 1993, Avignon, France.

[Garigliano, 1996]
R. Garigliano, "The Neopragmatist Manifesto", email: Roberto.Garigliano@durham.ac.uk, 1996.

[Glasgow, 1997] Glasgow, "The Glasgow Haskell Compiler", `http://www.dcs.gla.ac.uk/fp/-software/ghc/, 1997`.

[Grishman and Sundheim, 1995] R. Grishman and B. Sundheim, "Design of the MUC-6 Evaluation", in *Proceedings: Sixth Message Understanding Conference (MUC-6)*, pages 1–11, November 1995.

[Harman, 1996] D. Harman, "CHECK - Proceedings of TREC 4", Technical report, NIST, 1996.

[Heitz, 1996] J. Heitz, "An Investigation into Figurative Language in the LOLITA NLP System", Master's thesis, Department of Computer Science, University of Durham, 1996.

[Hindley and Seldin, 1986] J. Hindley and R. Seldin, *Introduction to combinators and lambda-calculus*, 1986.

[Hogg, 1997] J. Hogg, "Real World Applications of Functional Programming", `http://-www.dcs.gla.ac.uk/fp/realworld/, 1997`.

[Hopkins, 1993] M. Hopkins, "Demonstration of the Tomita Parsing Algorithm", `ftp://-iecc.com/pub/file/tomita.tar.gz or mark@omnifest.uwm.edu`, 1993.

[Hutchins, 1996] J. Hutchins, "Evaluation of Machine Translation and Translation Tools", `http://www.cse.ogi.edu/CSLU/HLTsurvey/ch13node5.html, 1996`.

[Jarvis, 1997] S. Jarvis, *Profiling Large-scale Lazy Functional Programs*, PhD thesis, LNLE, University of Durham, 1997.

[Kay, 1996] M. Kay, "Machine Translation: The Disappointing Past and Present", `http://-www.cse.ogi.edu/CSLU/HLTsurvey/ch8node4.html, 1996`.

[King, 1996] M. King, "Evaluating Natural Language Processing Systems", *CACM*, 39(1):73–79, 1996.

[Krupka, 1995] G. Krupka, "SRA: Description of the SRA System as Used for MUC-6", in *The Sixth Message Understanding Conference*, pages 221–236, Nov 1995.

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

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

[Morgan *et al.*, 1995] R. Morgan, R. Garigliano, P. Callaghan, S. Poria, M. Smith, A. Urbanowicz, R. Collingham, M. Costantino, C. Cooper, and the LOLITA Group., "Description of the LOLITA System as Used in MUC-6", in *The Sixth Message Understanding Conference*, pages 71–87, Nov 1995.

[Nettleton, 1995] D. J. Nettleton, *Evolutionary Algorithms in Artificial Intelligence: A Comparative Study Through Applications.*, PhD thesis, Department of Computer Science, University of Durham, 1995.

[Oepen *et al.*, forthcoming] S. Oepen, K. Netter, and J. Klein, "TSNLP - Test Suites for Natural Language Processing", in J. Nerbonne, editor, *Linguistic Databases*, forthcoming.

[Onyshkevych, 1993] B. Onyshkevych, "Template Design for Information Extraction", in *Proceedings: Fifth Message Understanding Conference (MUC-5)*, pages 19–23, August 1993.

[Pallett and Fourcin, 1996] D. S. Pallett and A. Fourcin, "Speech Input: Assessment and Evaluation", `http://www.cse.ogi.edu/CSLU/HLTsurvey/ch13node8.html, 1996`.

[Peterson and Hammond, 1997] J. Peterson and K. Hammond, "Haskell Report 1.3", `http://-haskell.cs.yale.edu/haskell-report/haskell-report.html, 1997`.

[Peyton Jones, 1996] S. L. Peyton Jones, "Bulk types with class", `http://-www.dcs.glasgow.ac.uk/fp/workshops/fpw96/Proceedings96.html`, July 1996.

[Pizza, 1997] Pizza, "The Pizza Group Home Page", `http://wwwipd.ira.uka.de/~pizza/, 1997`.

[European Commission, 1996] European Commission, "Language Engineering", `http://-www2.echo.lu/langeng/en/lehome.html, 1996`.

[Runciman and Wakeling, 1993] C. Runciman and D. Wakeling, "Heap Profiling of Lazy Functional Programs", Technical Report YCS-92-172, University of York, Computer Science Department, 1993.

[Sansom, 1994] P. M. Sansom, *Execution profiling for non-strict functional languages*, PhD thesis, Dept. of Computing Science, University of Glasgow, 1994.

[Santos, 1995] A. Santos, *Compilation by Transformation in Non-Strict Functional Languages*, PhD thesis, Department of Computing Science, University of Glasgow, 1995.

[Shiu, 1997] S. K. Y. Shiu, *Type Theoretic Semantics for Semantic Networks: An Application to Natural Language Engineering*, PhD thesis, Department of Computer Science, University of Durham, 1997.

[Sikorski and Allen, 1996] T. Sikorski and J. F. Allen, "TRAINS-95 System Evaluation", Technical Report 3, Univ. of Rochester, 1996.

[Smith, 1996] M. H. Smith, *Natural Language Generation in the LOLITA System: An Engineering Approach.*, PhD thesis, Department of Computer Science, University of Durham, 1996.

[Sparck Jones and Galliers, 1996] K. Sparck Jones and J. Galliers, *Evaluating Natural Language Processing Systems: An Analysis and Review*, Springer-Verlag, 1996.

[Sprent, 1989] P. Sprent, *Applied Nonparametric Statistical Methods*, Chapman and Hall, 1989.

[Steele, 1990] G. Steele, *Common Lisp : the language*, Bedford, 1990.

[Sun, 1997] Sun, "Sun Microsystems' Java Page", `http://www.sun.com/java/, 1997`.

[Sundheim, 1993] B. Sundheim, "Tipster/MUC-5 Information Extraction System Evaluation", in *Proceedings: Fifth Message Understanding Conference (MUC-5)*, pages 27–44, August 1993.

[Sundheim, 1995] B. Sundheim, "Overview of Results of the MUC-6 Evaluation", in *Proceedings: Sixth Message Understanding Conference (MUC-6)*, pages 13–31, November 1995.

[Sundheim, 1996] B. Sundheim, "Task-Oriented Text Analysis Evaluation", `http://-www.cse.ogi.edu/CSLU/HLTsurvey/ch13node4.html, 1996`.

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

[Trinder *et al.*, 1996] P. W. Trinder, K. Hammond, J. S. Mattson, A. S. Partridge, and S. L. Peyton Jones, "GUM: a portable parallel implementation of Haskell", `http://-www.dcs.gla.ac.uk/fp/authors/Philip_Trinder/gumFinal.ps.Z`, May 1996.

[Trinder *et al.*, 1997] P. Trinder, K. Hammond, H. Loidl, and S. L. Peyton Jones, "Algorithm + Strategy = Parallelism", *J. Functional Programming (submitted)*, 1997.

[Verbmobil, 1997] Verbmobil, "The Verbmobil Project", `http://www.dfki.uni-sb.de/-verbmobil/, 1997`.

[Vilain *et al.*, 1995] M. Vilain, J. Burger, J. Aberdeen, D. Connolly, and L. Hirschman", "A Model-Theoretic Coreference Scoring Scheme", in *Proceedings: Sixth Message Understanding Conference (MUC-6)*, pages 45–52, November 1995.

[Wadler, 1995] P. Wadler, "Monads for functional programming", in J. Jeuring and E. Meijer, editors, *Advanced Functional Programming*, volume 925 of *LNCS*, Springer Verlag, 1995.

[Wang, 1994] Y. Wang, *An Intelligent Computer-based Tutoring Approach for the Management of Negative Transfer*, PhD thesis, Department of Computer Science, Durham University, 1994.

[Zaenen, 1996] A. Zaenen, "Multilinguality", `http://www.cse.ogi.edu/CSLU/HLTsurvey/-ch8node2.html, 1996`.