Evaluating an Information Extraction System

W. Lehnert, C. Cardie, D. Fisher, J. McCarthy, E. Riloff, & S. Soderland*

COMPUTER SCIENCE DEPARTMENT, LGRC
UNIVERSITY OF MASSACHUSETTS
BOX 34610
AMHERST MA 01003-4610
lehnert@cs.umass.edu

*University of California, Berkeley
Department of Computer Science
Berkeley, CA

Abstract:

Many natural language researchers are now turning their attention to a relatively new task orientation known as *information extraction.* Information extraction systems are predicated on an I/O orientation that makes it possible to conduct formal evaluations and meaningful cross-system comparisons. This paper presents the challenge of information extraction and shows how information extraction systems are currently being evaluated. We describe a specific system developed at the University of Massachusetts, identify key research issues of general interest, and conclude with some observations about the role of performance evaluations as a stimulus for basic research.

1. Information Extraction from Text

Progress in natural language processing is typically divided into major subareas according to the types of issues being addressed. Methods for sentence analysis associated with syntactic structures and full sentence grammars are strongly influenced by theoretical linguistics. In-depth narrative text comprehension is often pursued by psychologists and others with an interest in human memory models. Natural language interfaces for practical database access are influenced by the logical structure of formal query languages and relational databases. Work on discourse analysis builds its foundations on formalisms for planning and user modelling.

Very recently a new orientation for research in natural language processing has emerged under the name of *information extraction*. This area addresses information processing needs associated with large volumes of text containing information from some domain of interest. For example, a stock analyst might want to track news stories about corporate mergers; an intelligence analyst might need to track descriptions of terrorist events in a geographic region; an insurance adjuster might want to compile data from text-based hospital records. In general, information extraction refers to the problem of finding useful information in a collection of texts, and encoding that information in a format suitable for incorporation into a database.

Acknowledgements: This research was supported by Office of Naval Research Contract N00014-92-J-1427 and NSF Grant no. EEC-9209623, State/Industry/University Cooperative Research on Intelligent Information Retrieval.

At the present time, most information extraction tasks are managed manually, by human analysts who read the source documents and create database entries by hand. This work is time-consuming, tedious, and difficult to monitor for quality control. In recent years, a growing number of researchers in natural language processing have begun to look at information extraction applications and organize entire research projects around the special requirements of this relatively new task orientation. As a result, we are coming to understand that the language processing requirements of information extraction are somewhat different from the requirements of other language processing applications. Furthermore, a body of existing technology is already positioned to address the information extraction problem with promising results.

The information extraction problem is especially interesting as a subarea in natural language processing insofar as it lends itself to comparative performance evaluations and relatively clean evaluation metrics. Because the output of an information extraction system is a populated database, it is possible to compare the output of a given system to an ideal database in order to determine how well that system is doing. This distinguishes information extraction systems from other natural language processing systems where evaluation is highly problematic.

In this paper we will overview two recent performance evaluations in information extraction, and describe an information extraction system developed at the University of Massachusetts. We will present a detailed evaluation of our own system along with an extensive example of information extraction in action. In closing, we will discuss the limitations of quantitative evaluations and the importance of qualitative assessments during system development.

2. A First Message Understanding Evaluation (MUC-3)

In 1991 DARPA sponsored an ambitious evaluation of information extraction technologies in the form of the Third Message Understanding Conference (MUC-3). This was a first attempt at a quantitative performance evaluation based on blind test sets and rigorous test procedures. Fifteen research labs participated in MUC-3 and a published proceedings records both test results and system descriptions [Sundheim 1991]. The process of coordinating fifteen laboratories in an evaluation of large complicated language processing systems is a difficult one, but MUC-3 was conducted with considerable planning and foresight.[1]

The MUC-3 performance evaluation utilized a development corpus of 1300 documents describing terrorist activities in Latin America.[2] Each text in the development corpus was paired with hand-coded template instantiations designed to capture all relevant information embodied in the source text. The MUC-3 challenge was to design a system that could generate these template instantiations automatically, without any human assistance. Each MUC-3 system was evaluated on the basis of a blind test set consisting of 100 new documents. No alterations to the participating systems were allowed once a

[1] We note that earlier attempts at comparative performance evaluation took place at the MUCK1 and MUCK2 message understanding conferences in 1987 and 1989. Although these earlier attempts were less ambitious in their scope than MUC-3, they created a foundation that was instrumental to the success of the 1991 evaluation.

[2] All text materials were provided by the Foreign Broadcast Information Service.

site had downloaded the test materials for the final evaluation. The results were scored by a program that compared system-generated "response" templates to hand-coded "key" templates (see Figure 1).

Figure 1: A Method for Evaluating Message Understanding Systems
Figure 1 is not available at this time.

A serious description of the MUC-3 tests and evaluation becomes fairly involved and cannot be covered in the space of a few paragraphs. For our purposes we'll show some sample input, some sample output, and a sample score report. A more comprehensive overview can be found in [Lehnert and Sundheim 1991].

Here is a sample text taken from the first blind test set (TST1):

```
TST1-MUC3-0004

  BOGOTA, 30 AUG 89 (INRAVISION TELEVISION CADENA 2) -- [TEXT] LAST NIGHT'S
TERRORIST TARGET WAS THE ANTIOQUIA LIQUEUR PLANT. FOUR POWERFUL ROCKETS WERE
GOING TO EXPLODE VERY CLOSE TO THE TANKS WHERE 300,000 GALLONS OF THE SO-
CALLED CASTILLE CRUDE, USED TO OPERATE THE BOILERS, IS STORED. THE WATCHMEN
ON DUTY REPORTED THAT AT 2030 THEY SAW A MAN AND A WOMAN LEAVING A SMALL
SUITCASE NEAR THE FENCE THAT SURROUNDS THE PLANT. THE WATCHMEN EXCHANGED FIRE
WITH THE TERRORISTS WHO FLED LEAVING BEHIND THE EXPLOSIVE MATERIAL THAT ALSO
INCLUDED DYNAMITE AND GRENADE ROCKET LAUNCHERS, METROPOLITAN POLICE PERSONNEL
SPECIALIZING IN EXPLOSIVES, DEFUSED THE ROCKETS. SOME 100 PEOPLE WERE WORKING
INSIDE THE PLANT.

  THE DAMAGE THE ROCKETS WOULD HAVE CAUSED HAD THEY BEEN ACTIVATED CANNOT BE
ESTIMATED BECAUSE THE CARIBE SODA FACTORY AND THE GUAYABAL RESIDENTIAL AREA
WOULD HAVE ALSO BEEN AFFECTED.

  THE ANTIOQUIA LIQUEUR PLANT HAS RECEIVED THREATS IN THE PAST AND MAXIMUM
SECURITY HAS ALWAYS BEEN PRACTICED IN THE AREA. SECURITY WAS STEPPED UP LAST
NIGHT AFTER THE INCIDENT. THE LIQUEUR INDUSTRY IS THE LARGEST FOREIGN
EXCHANGE PRODUCER FOR THE DEPARTMENT.
```

The relevant information in each text may have one key template, more than one key template, or no key template if the text is deemed to be irrelevant. The text given above has one key template associated with it (see Figure 2).

The scoring program evaluates overall system performance by checking each response template against the available key templates. Information must be properly positioned in the right slots and in the right response templates in order to be counted correct. The *Recall* score measures the ratio of correct information extracted from the texts against all the available information present in the texts. The *Precision* score measures the ratio of correct information that was extracted against all the information that was extracted.[3] For a test run of 100 texts, recall and precision are

[3] For example, suppose you answered two of four test questions correctly. Your recall score for the test would be 50%. But your precision score for the test would depend on how many questions you answered altogether. If you answered all four, your precision would be 50%. If you answered three questions, your precision would be 75%, and if you were smart enough to only answer two questions, then your precision would be 100%.

averaged over all the output templates and computed four different ways. Figure 3 shows the official UMass score report from the final MUC-3 test run on a second set of 100 blind texts (TST2).

```
0. MESSAGE ID                     TST1-MUC3-0004
1. TEMPLATE ID                    1
2. DATE OF INCIDENT               29 AUG 89
3. TYPE OF INCIDENT               ATTEMPTED BOMBING
4. CATEGORY OF INCIDENT           TERRORIST ACT
5. PERPETRATOR: ID OF INDIV(S)    "MAN"
                                  "WOMAN"
6. PERPETRATOR: ID OF ORG(S)      -
7. PERPETRATOR: CONFIDENCE        -
8. PHYSICAL TARGET: ID(S)         "ANTIOQUIA LIQUEUR PLANT"/
                                  "LIQUEUR PLANT"
9. PHYSICAL TARGET: TOTAL NUM     1
10. PHYSICAL TARGET: TYPE(S)      COMMERCIAL: "ANTIOQUIA LIQUEUR PLANT"/
                                              "LIQUEUR PLANT"
11. HUMAN TARGET: ID(S)           "PEOPLE"
12. HUMAN TARGET: TOTAL NUM       PLURAL
13. HUMAN TARGET: TYPE(S)         CIVILIAN: "PEOPLE"
14. TARGET: FOREIGN NATION(S)     -
15. INSTRUMENT: TYPE(S)           *
16. LOCATION OF INCIDENT          COLOMBIA: ANTIOQUIA (DEPARTMENT)
17. EFFECT ON PHYSICAL TARGET(S)  NO DAMAGE: "ANTIOQUIA LIQUEUR PLANT"/
                                             "LIQUEUR PLANT"
18. EFFECT ON HUMAN TARGET(S)     NO INJURY OR DEATH: "PEOPLE"
```

Figure 2: A Sample Answer Key

Figure 3: The Official UMass Score Report from MUC-3
Figure 3 is not available at this time

Although four scoring metrics were computed for each score report, MATCHED/MISSING was designated as the official scoring metric for MUC-3. Figure 4 shows a scatter plot for the recall and precision of all 15 sites under the official scoring metric.

Figure 4: Overall Recall/Precision Results for all the MUC-3 sites
Figure 4 is not available at this time.

It is easy to drown in the numbers of these score reports, but it is important to look beyond the numbers and the scatter plots at the bigger picture. The development effort behind the UMass system yielded three important results, and only one of them is evident in the MUC-3 score reports:

(1) The UMass/MUC-3 system was relatively successful. UMass posted the highest combined scores for recall and precision of all the systems tested.

(2) The UMass/MUC-3 system was relatively expensive to build. We estimated that 2.25 person/years of effort went into MUC-3. This represents more effort than any of the other sites would admit to, and twice as much as the average effort level underlying all the MUC-3 systems.

(3) The graduate students who implemented the UMass/MUC-3 system had no desire to ever build anything like it again. Their labor was time consuming and tedious. They established the viability of the UMass approach relative to other approaches, but with a human labor factor that threw into question the practicality of the technology.

Our participation in MUC-3 benefited our research in a variety of ways. On the one hand, it established the viability of our general approach to information extraction. On the other hand, it presented us with a major research challenge that could not be ignored. The difference between a viable message understanding technology and a practical message understanding technology lies in the ease with which that technology can be ported across domains. In order to examine this issue more substantively, we need to move beyond the score reports and the scatter plots and look at the specifics of the system that UMass ran at MUC-3.

3. The CIRCUS Sentence Analyzer

CIRCUS is a conceptual analyzer that produces semantic case frame representations for input sentences. Although space does not permit us to give a full technical description of CIRCUS, we will attempt to convey some sense of sentence analysis via CIRCUS. For more details, please consult [Lehnert 1991; and Cardie and Lehnert 1991].

CIRCUS uses no syntactic grammar and produces no parse tree as it analyzes a sentence. Rather, it uses lexically-indexed syntactic knowledge to segment incoming text into noun phrases, prepositional phrases, and verb phrases. These constituents are stored in global buffers that track the subjects, verbs, direct objects, and prepositional phrases of a sentence. Because we restrict the buffer contents to simple constituents with a highly local sense of the sentence, larger constituents like clauses are not explicitly stored by the syntactic component of CIRCUS.

While syntactic buffers are being filled with sentence fragments, a mechanism for handling predictive semantics is responsible for establishing case role assignments. Semantic case frames are activated by concept node (CN) definitions, and each CN definition can be triggered by one or more lexical items. These CN triggers index the CN definitions and initiate all CN-related processing. Associated with each slot in a CN are both hard and soft constraints. A hard constraint is a predicate that *must* be satisfied, while a soft constraint defines a *preference* rather than an absolute requirement. When a CN instantiation meets certain criteria established by the CN definition, CIRCUS freezes that case frame and passes it along as output from the sentence analyzer. A single sentence can generate an arbitrary number of case frame instantiations depending on the conceptual complexity of the sentence and the availability of relevant CN definitions in the CN dictionary.

Because CIRCUS is designed to generate case frame representations in response to sentence fragments, ungrammatical sentences or sentences with highly complicated syntactic structures are often navigated without difficulty. CIRCUS was designed to maximize robust processing in the face of incomplete knowledge. It does not require

complete dictionary coverage with respect to CN definitions or even part-of-speech recognition, so CIRCUS is especially well-suited for information extraction applications from unconstrained text. As described above, the first serious evaluation of CIRCUS took place with MUC-3, where CIRCUS posted the highest combined scores for recall and precision of all the participating sites [Lehnert et al. 1991a, 1991b, and 1991c]. However, questions were raised about the portability and scalability of CIRCUS' approach to information extraction.

We welcomed an opportunity to focus on practicality issues in preparing for the next message understanding performance evaluation in 1992. The dictionary used by the UMass/MUC-3 system emerged after roughly 1500 hours of highly skilled labor by two advanced graduate students and one post doc. This one aspect of the system represented the single largest investment of manual knowledge engineering and the greatest bottleneck with respect to portability across domains. A practical information extraction system would require some capability for automated dictionary construction in order to reduce this manual engineering requirement. Although the difference between a viable dictionary and a practical dictionary is not one that can be seen in any score reports, we decided to focus on this aspect of our system development because it represented a compelling research issue. We believe that it is important to participate in performance evaluations, but it is equally important to maintain a sense of the basic research issues, even if those issues do not lend themselves to the kinds of visible progress that can be measured in score reports.

4. A Second Message Understanding Evaluation (MUC-4)

The MUC-4 evaluation was conducted in 1992 with a format very similar to that used in MUC-3. A total of 19 organizations participated in the development of the MUC-4 systems, including 12 "veterans" of MUC-3 and five new university labs. The development corpus of Latin American terrorism was used again, but this time there were two test sets of 100 novel texts (TST3 and TST4) instead of one. A different scoring metric was chosen as the official metric (All Templates) because it was more sensitive to spurious templates than the metric used in MUC-3 (Matched/Missing). In addition to the template instantiation task, all systems were evaluated for their text filtering capabilities as well (a natural side effect of the information extraction task). A method for analyzing the statistical significance of the test results was also utilized [Chinchor et al 1993]. Because MUC-4 used the same domain as MUC-3, veteran sites were able to conduct system development experiments that picked up where their MUC-3 effort left off. For UMass, this meant addressing the questions that had been raised about portability and scalability [Lehnert et al 1992a and 1992b].

In confronting the issue of efficient system development, we chose to focus on the problem of dictionary construction. The dictionary used by UMass/MUC-3 emerged after roughly 1500 hours of highly skilled labor by two advanced graduate students and one post doc. For MUC-4, we created a new dictionary that achieved nearly the full functionality of the UMass/MUC-3 dictionary after only 8 hours of effort on the part of a first-year graduate student. This outcome was achieved through the use of an automated dictionary construction tool called AutoSlog [Riloff and Lehnert 1993, Riloff 1993a].

We ran two versions of our system for MUC-4. The official system and a second optional system were identical except for their dictionaries. Both dictionaries accessed the same 5436 lexical definitions for part-of-speech recognition and word senses (these definitions were taken from the UMass/MUC-3 dictionary), along with

2102 proper names. However, the optional system ran a CN dictionary constructed by AutoSlog containing 379 concept node definitions. Our official system ran a version of the UMass/MUC-3 CN dictionary augmented by 76 additional concept node definitions imported from the AutoSlog dictionary for a total of 389 concept node definitions. Prior to the official test runs, we predicted that both systems would produce comparable levels of precision, and that the optional system would fall behind the official system by 10 recall points under All Templates. Table I contains the All Templates scores for all four test runs.[4]

System	recall	precision	P&R	2P&R	P&2R
Official TST3	47	57	51.52	54.67	48.71
Optional TST3	40	59	47.67	53.88	42.75
Official TST4	46	42	43.91	42.74	45.14
Optional TST4	36	46	40.39	43.58	37.64

Table I: Overall Scores for Four Test Runs

As predicted, the AutoSlog dictionary produced lower recall levels than the official system: 7 points lower for TST3, and 10 points lower for TST4. Precision was comparable for both systems with AutoSlog generating higher overall precision rates: 2 points higher for TST3 and 4 points higher for TST4. We note that our performance on TST4 was generally worse than TST3. However, a close inspection of the detailed score reports for the official system shows that the primary difference in those reports lies in the All Templates precision scores: 57 for TST3 vs. 42 for TST4. This loss of precision can be explained for the most part by comparing the number of spurious templates: 16 for TST3 vs. 34 for TST4.

Setting aside the differences between TST3 and TST4, we were pleased to see how well the AutoSlog dictionary performed relative to our hand-crafted dictionary from the previous year. Comparing P&R scores, our AutoSlog dictionary achieved 93% of the overall performance of our official system on TST3, and 92% of the official system's performance on TST4. In an effort to leverage the UMass/MUC-3 and the AutoSlog dictionaries, we strengthened the performance of the MUC-3 dictionary by augmenting it with 76 AutoSlog definitions.[5] This was done because our official system was designed to demonstrate our best capabilities with respect to information extraction. A cleaner comparison of our original MUC-3 dictionary vs. the AutoSlog dictionary would have resulted in a smaller difference in overall performance, and given AutoSlog a much stronger showing. As it stands, we are comparing the pure AutoSlog dictionary in our optional runs to the official run dictionary that derived 20% of its definitions from AutoSlog.

Our MUC-4 test results have demonstrated that an effective domain-dependent dictionary can be efficiently constructed using a representative text corpus

[4] A single scoring measure known as the F-measure was used to rank the MUC-4 systems. Under P&R, the F-measure weighed recall and precision equally. Under 2P&R, the measure favored precision over recall, and under P&2R the measure favored recall over precision.

[5] Other methods of combining the two dictionaries were tested, but this was the most effective strategy.

accompanied by hand-coded template encodings. Our preliminary and very limited efforts produced a dictionary that closely mirrors the functionality obtained by a relatively successful hand-crafted dictionary. Although the process of dictionary construction via AutoSlog is not completely automated, the manual labor needed can be completed in a matter of hours by a single individual with minimal expertise in dictionary construction. As a result, we are now in a position to claim that effective customized dictionaries can be constructed quickly and easily by relatively inexperienced system developers. We consider this to be a significant step forward in the area of automated dictionary construction for text extraction applications.

5. Automated Dictionary Construction

The AutoSlog construction tool analyzes available key templates in conjunction with source texts and generates hypothesized CIRCUS definitions without human assistance. AutoSlog's proposed concept node definitions are derived from sentences in the MUC-4 development corpus that contain string fills found in their associated key templates. Using the complete 1300-text DEV corpus as the training set for our dictionary construction experiment, AutoSlog proposed 1356 concept node definitions based on 1272 string-fill slots. Although a large number of these definitions were flawed in some way, 375 of the 1356 definitions (28%) proposed by AutoSlog were deemed acceptable when reviewed by visual inspection.

Each AutoSlog definition begins with a single string fill in a single key template. Given a specific slot, AutoSlog extracts the first non-empty string fill listed in the key template (string-fill slots often contain multiple strings based on multiple references within the source text). It then searches the source text for the first instance of that string within the source text. Once located, AutoSlog pulls the complete sentence containing that string from the source text and passes it to the CIRCUS sentence analyzer for syntactic analysis. CIRCUS analyzes the sentence using a part-of-speech dictionary. If successful, a set of buffers is instantiated with simple syntactic constituents corresponding to a subject, a verb, and possibly an object or a prepositional phrase. When the original string fill shows up in one of these buffers, AutoSlog hypothesizes a concept node definition complete with a lexical trigger, complement pattern, and slot constraints. This definition is then written to a file and AutoSlog returns to the key template for the next string fill slot. Figure 5 shows the AutoSlog construction tool in action.

Figure 5: Automated Dictionary Construction
Figure 5 is not available at this time.

The presence of string-fills in key templates is a crucial requirement for AutoSlog. In fact, the more string-fill slots, the better. Each MUC-4 template contained six string-fill slots used by AutoSlog: inc-instr-id, perp-ind-id, perp-org-id, phys-tgt-id, hum-tgt-name, and hum-tgt-desc. After processing the 1300 texts of DEV, AutoSlog generated 136 definitions based on inc-instr-id, 316 definitions from perp-ind-id, 201 definitions from perp-org-id, 306 definitions from phys-tgt-id, 193 definitions from hum-tgt-name, and 204 definitions from hum-tgt-desc. This dictionary was compiled in a 14-hour batch run and then passed to a CIRCUS programmer for manual review. During the review process, each definition was dispatched into one of two possible states: (1) keep as is, or (2) save for possible revision. Files were maintained for the "keeps" and the "edits" with the expectation that the keep-definitions might be

augmented by some number of edit-definitions that could be salvaged via manual knowledge engineering. The initial categorization into "keeps" and "edits" was relatively fast because each definition could be categorized on the basis of visual inspection alone. Many definitions destined for the edit files were easy to spot since they often resulted from parsing errors, patterns of no linguistic generality, or patterns of dubious reliability.

Here is an example a good AutoSlog definition generated by the first text in the development corpus:

```
---------------------------------------------------------------------------------------------------------------------------------
Id: DEV-MUC3-0001      Trigger: KIDNAPPED     Trigger Root: KIDNAP     Syntactic-type: VERB
Slot filler: "TERRORISTS"
Sentence: (THE ARCE BATTALION COMMAND HAS REPORTED THAT ABOUT &&50 PEASANTS OF
          VARIOUS AGES HAVE BEEN KIDNAPPED BY TERRORISTS OF THE
          FARABUNDO_MARTI_NATIONAL_LIBERATION_FRONT IN SAN_MIGUEL DEPARTMENT >PE)
Name: %ACTOR-PASSIVE-VERB-PP-KIDNAPPED-BY%
Time limit: 10
Variable Slots:                          Constraints:
(ACTOR  (*PP* (IS-PREP? '(BY))))             (((CLASS ORGANIZATION *PP*)
                                              (CLASS TERRORIST *PP*)
                                              (CLASS PROPER-NAME *PP*)
                                              (CLASS HUMAN *PP)))
Constant Slots: (TYPE PERPETRATOR)
Enabling Conditions: ((PASSIVE))
---------------------------------------------------------------------------------------------------------------------------------
```

This definition extracts slot fillers from constructions of the form: "X is/was/(has/have been) kidnapped by Y." This particular definition will only pick up the conceptual actor Y. A separate definition is needed to pick up the conceptual victim X. The following is an example of a bad AutoSlog definition:

```
---------------------------------------------------------------------------------------------------------------------------------
Id: DEV-MUC3-0036      Trigger: WAS           Trigger Root: WAS        Syntactic-type: VERB
Slot Filler: "MEMBER OF THE DEMOCRATIC SOCIALIST PARTY"
Sentence: (GILDA FLORES WAS AN ACTIVE MEMBER OF THE DEMOCRATIC SOCIALIST PARTY OF
          GUATEMALA >CO WHOSE SECRETARY_GENERAL MARIO SOLORZANO REPORTED
          SALVADORAN PARAMILITARY GROUPS ARE CARRYING_OUT ACTIONS IN THIS COUNTRY
          >PE)
Name: %VICTIM-ACTIVE-OBJECT-VERB-WAS%
Time Limit: 10
Variable Slots:                          Constraints:
(VICTIM (*DO* 1))                            (((CLASS HUMAN *DO*)
                                              (CLASS PROPER-NAME *DO*)))
Constant Slots: (TYPE KIDNAPPING)
Enabling Conditions: ((ACTIVE :CHECK-DO-NO-ONE T))
---------------------------------------------------------------------------------------------------------------------------------
```

As it stands, this definition hypothesizes that the verb “was” predicts the victim of a kidnapping. Although the source sentence does legitimately suggest that the verb “to be” can be used to link human names with human descriptions, this proposed definition cannot be trusted to deliver a kidnapping victim.

When AutoSlog creates a new definition, it checks the existing set of previously proposed definitions to see if the current proposal duplicates an older one. AutoSlog does not produce multiple copies of the same definition. By tracking the number of duplicates AutoSlog suppresses, we can see evidence that the dictionary is approaching a saturation point. In particular, we note that after AutoSlog has

processed 1200 texts, nearly two thirds of its proposed definitions for the next 100 texts are duplicates of previously generated definitions. Figure 6 shows the weakening frequency of new dictionary definitions as we move through the development corpus.

Figure 6: Dictionary Saturation Under AutoSlog
Figure 6 is not available at this time.

Although the AutoSlog dictionary definitions are derived from only six template slots, template-generation routines are capable of extracting the information needed to fill additional slots. When the AutoSlog dictionary operates in conjunction with the full system, we can fill every template slot except phys-tgt-effect, phys-tgt-total-num, and hum-tgt-total-num.

The 8-hour AutoSlog dictionary was completed only four weeks before the final testing for MUC-4. After seeing how much impact AutoSlog can have on the process of dictionary construction, we are now pursuing enhancements to AutoSlog in order to strengthen its baseline performance. As it stands, AutoSlog can be moved to new domains with a minimal amount of software tuning. Adjustments must be made to handle a new template format, but any templates that contain string-fills will serve to fuel dictionary construction.

6. Looking Under all the Numbers

As a result of our experience with MUC-3 and MUC-4 we believe we have learned how to best use a performance evaluation in our own system development effort. When serious effort goes into the analysis of extensive score reports that utilize different metrics for measuring performance in a variety of ways, it is easy to be seduced by all the numbers and lose sight of less rigorous feedback mechanisms. We have conducted a post hoc analysis of our system's performance on TST3 in order to better understand the various problems encountered on TST3. Most of this analysis explores the behavior of CIRCUS, its use of concept node definitions, and the effects of memory-based consolidation.[6] As detailed and useful as the score reports are, they are not designed to tease apart the individual performance contributions of a sentence analyzer, discourse analyzer, or template generator. Subcomponents like these must be analyzed separately if we want to understand where to focus future development efforts.

In order to understand some of the discussion to follow, it will help to know what is meant by “mapping problems” during scoring. Whenever there is one response template and exactly one key template, there can be no mapping problem. The lone response template is compared to the key template and scoring proceeds on a slot-by-slot basis. But suppose the number of response templates does not correspond to the number of key templates. Then we can have a mapping problem.

For example, suppose there are two response templates and only one key template. According to the scoring guidelines, only one response template can be compared to the key template. The remaining response template is deemed “spurious” and all of the information it contains will be counted as spurious information, even if it appears

[6] Memory-based consolidation refers to that part of the system that analyzes concept node instantiations generated by CIRCUS and maps them into template instantiations of the type shown in figure 2. More will be said about memory-based consolidation at the end of this section.

in the key template. So which response template should be mapped to the key template and which should be counted as spurious? In general, we would like to make this decision in a way that maximizes our overall score.

The scoring program attempts to solve the mapping problem via a hill-climbing algorithm, but hill-climbing does not guarantee optimal solutions, so the scoring program can make mapping errors. It is possible for humans to intervene with mapping specifications for specific situations, but mapping problems can be very difficult to resolve by visual inspection, and recall/precision trade offs are common.

When "good" information is placed in a spurious template or a template that doesn't map to the key containing that information, we lose both recall and precision. Some of the scoring metrics are more forgiving with respect to precision losses due to spurious templates (e.g. MATCHED/ONLY is forgiving while ALL TEMPLATES is not), but all of the scoring metrics register recall losses associated with spurious templates.

So we see that it is not enough to craft a system that extracts information perfectly at the sentence level. It is also critical that the information be grouped into the response templates correctly with the right number of templates being generated. Human encoders do not always agree on the right number of templates for some of the more complicated documents, so it is understandable that the computer systems run into template mapping problems as well.

6.1 Recall Limitations

The dictionary used by the official UMass/MUC-4 system contained 389 concept node definitions. Of these, 172 (44%) were enabled[7] to process TST3. On average, each definition was enabled nearly 9 times for a total of 1515 concept node enablements (~15 per text on average).

For TST3, CIRCUS extracted 943 strings to fill variable slots in enabled concept nodes. Because of redundant concept node definitions, almost half of these string fills were duplicates, leaving 520 unique string fills extracted by CIRCUS. According to our analysis, 214 of these string fills were discarded during consolidation and 306 string fills made it into a response template.

Of the 520 non-redundant string-fills, 38% were correctly incorporated into a response template where they matched the string or strings listed in a key template. A full 34% were correctly discarded or merged by consolidation (and therefore did not make it into a response template). The sum of these instances accounts for 72% of the total string fills - all handled correctly by consolidation. Of the remaining string fills, 21% appeared in response templates as spurious slot fills, and 7% were incorrectly discarded. Of the 237 string fills that did legitimately correspond to slot fills in key templates, consolidation correctly incorporated 199 (84%) into response templates. Even so, our overall recall score was only 46%. Where are the other string fills?

Our analysis shows that CIRCUS generated 225 good (full match) string fills and 12 partially good (partial match) string fills. According to the score report, there were 416 possible string fills for TST3. That tells us CIRCUS is producing only 55% of the

[7] An enabled concept node is one that produces a case frame instantiation. All output generated by CIRCUS is based on enabled concept nodes.

possible string fills for TST3. This 55% hit rate effectively imposes a rough ceiling on our overall recall, and suggests that significant gains in recall will require stronger performance levels during sentence analysis.

6.2 Precision Limitations

When we examine the 306 string fills present in our TST3 response templates, we find that 187 could be matched to slot fills in some key template and 12 could be partially matched to a key template slot fill. If all of these string fills were in correct slots and correct templates, our string fill precision would be 63%. But only 142 string fills result in full matches with key template slots and 24 result in partial matches. Of the 107 strings that can't be matched to any key template, we have found the following breakdown of errors:

49 (46%) should have been discarded as irrelevant
16 (15%) were from mis-fired concept node definitions
15 (14%) were from parser errors
14 (13%) should have been merged with a more specific string
12 (11%) were from words not covered adequately by the dictionary
1 (1%) was from a source string altered by preprocessing

Of the 49 false hits associated with relevancy discriminations, our single greatest precision error came from 30 false hits on military clashes[8]. After that, four general problem areas share roughly equal responsibility for a significant number of errors: (1) false hits associated with faulty concept node definitions, (2) CIRCUS sentence analysis errors, (3) consolidation merging failures, and (4) inadequate dictionary coverage.

6.3 Dictionary Coverage

Although inadequate dictionary coverage was identified as a source of visible precision loss, we have been remarkably well-served by a relatively small dictionary of 5436 lexical items augmented by 2102 proper names. An analysis of the TST3 lexicon shows that 1008 words appearing in TST3 were not recognized by our system. Of these, 696 occurred only once. Of the remaining 312 words, the vast majority were proper names. A visual inspection of the unrecognized word list suggested that our lexicon was apparently adequate for the demands of TST3. However, this does not mean that all of our associated definitions were above reproach.

Our official dictionary contains a total of 389 concept node definitions, but only 172 of these were used during TST3. A frequency analysis of these 172 definitions showed that 20% of the definitions generated 74% of the string fills. A total of 37 (22%) concept node definitions failed to produce any string fills (perhaps these contain no variable slots or maybe they were discarded during consolidation), while one concept node definition produced 65 string fills, and three others produced over 50 string fills (each). Figure 7 shows the complete frequency distribution.

Figure 7: Concept Node Frequency Distribution
Figure 7 is not available at this time.

[8]A military clash was defined to be fighting between two military organizations, and these incidents were not designated as terrorist incidents according to our domain definition.

6.4 Comparing TST3 and TST4

Our F-scores suggest dramatic differences between TST3 and TST4, but a closer look suggests that there is just one underlying factor responsible for these divergent score summaries. Table II shows a more detailed perspective on the differences and similarities between TST3 and TST4.

	----------TST3----------			----------TST4----------		
	REC	PRE	OVG	REC	PRE	OVG
MATCHED/MISSING	47	67	14	46	68	14
MATCHED/SPURIOUS	60	57	26	63	42	46
MATCHED/ONLY	60	67	14	63	68	14
ALL TEMPLATES	47	57	26	46	42	46
SET FILLS ONLY	50	71	13	49	73	14
STRING FILLS ONLY	37	57	20	42	64	17
F-SCORES	51.52	54.67	48.71	43.91	42.74	45.14

Table II: TST3 and TST4 score reports from the official test runs

Reviewing the scores in Table II, we see that there is a remarkable similarity across most of the scores for TST3 and TST4. Our TST4 score was even better than our TST3 score under String Fills Only. The major differences appear in the precision and overgeneration scores for Matched/Spurious and All Templates. These differences also correspond to a striking difference in the number of spurious templates generated for TST3 (16) and TST4 (34). Looking deeper into the problem, we determined that two factors seemed to contribute to the large number of spurious templates in TST4.

First, many legitimate templates were deemed spurious because of mapping problems. We can see some evidence of this in the results of a comparative test designed to assess the impact of templates lost due to incorrect incident-type slot fills. In the comparative test, we will use "ATTACK" as the slot fill for all incident-type slots. This ensures that no template is deemed spurious due to a mapping restriction based on incident types. Table III shows the resulting F scores from comparative test runs for both TST3 and TST4, running the official UMass/MUC-4 system and generating batch score reports throughout.

	P&R	2P&R	P&2R
TST3 - official system	46.47	49.64	43.68
TST3 - (mapping restriction circumvented)	45.46	48.63	42.67
TST4 - official system	39.44	38.56	40.36
TST4 - (mapping restriction circumvented)	40.98	40.38	41.58

Table III: The Effect of Spurious Templates Lost to Incorrect Incident-Types

In comparing the net effect of the "all attacks" heuristic, we find that there is no advantage to any of the F-scores when all templates are typed as attacks in TST3. Indeed, there is a uniform drop of one point across the board when "all attacks" is turned on. On the other hand, the F scores for TST4 all benefit from the "all attacks" heuristic. P&R goes up 1.54, 2P&R goes up 1.82, and P&2R goes up 1.22. This tells us that we did not tend to lose otherwise legitimate templates because of their incident types in TST3, whereas a significant number of legitimate templates were lost for this reason in TST4. Precision is most dramatically affected by these errors, but P&R may have lost at least 2 points because of this problem.

Second, a large number of spurious templates were created when military targets were not recognized to be military in nature[9]. We have already seen how military clashes were the single greatest source of spurious string fills in TST3. Even so, we generated only 3 spurious templates due to false hits on military targets in TST3. In TST4 we generated 12 spurious templates for the same reason. If we assume that each spurious template contains at least 10 slot fills (a reasonable assumption when template filtering is in place), it follows that 120 spurious slot fills are coming from false hits on military targets in TST4. Removing 120 spurious slot fills from the TST4 score report, the precision score under All Templates goes from 42 to 48, and the P&R score goes up about 3 points as a result.

6.5 Memory-Based Consolidation

In general, we do not expect CIRCUS to filter out spurious information. It seems more appropriate for the sentence analyzer to pick up all events describing terrorist-related events, and then let a subsequent processor apply any domain-specific filters designed to recognize illegitimate targets. This separation of functionality allows CIRCUS to work with reasonably generic concept node definitions (albeit a set of concept node definitions that are determined by the domain). In our MUC-3 and MUC-4 systems, a consolidation module was used to manage intersentential phenomena such as pronoun resolution, discourse analysis, and additional filtering required by the domain definition. More generally, consolidation refers to the problem of mapping CN instantiations produced by CIRCUS into event descriptions appropriate for template instantiations. Since information pertaining to a single event can be distributed across multiple sentences, problems associated with consolidation are challenging, especially when a text describes multiple events. It is necessary to know when different noun phrases point to the same referent and when the topic shifts from one event to another.

The UMass/MUC-3 system used a rule based consolidation module which was largely dominated by rules designed to merge appropriate structures. Because the rule base was large (168 rules), it was difficult to pinpoint weak spots in the rule base and it became increasingly difficult to make reliable adjustments as needed. Our dissatisfaction with the UMass/MUC-3 approach prompted us to design a new consolidation module for MUC-4.

The UMass/MUC-4 consolidation module is "memory-based" in the sense that it assumes a specific memory organization strategy, and all processing is motivated by a small number of memory manipulations. The basic structure of memory-based

[9]According to our domain guidelines, an attack on a military target does not describe a terrorist incident.

consolidation (MBC) is a simple stack of incident structures, along with two associated stacks that track human targets and physical targets. At the end of each consolidation run, the number of incident structures on the incident stack usually corresponds to the number of templates we will instantiate, with each incident structure containing all the information needed to fill at least one template.

The incident structure serves as the basic data type inside MBC as well as the data type that is output from MBC. An incident structure is a frame consisting of slots for a date, location, perpetrators, and subevents. Each subevent consists of a specific incident type (murder, bombing, robbery, etc.) along with victims, physical targets, instruments, and effects. Although multiple subevents are permitted in an incident-structure to handle combined events like an arson/robbery combination, most incident structures contain only one subevent. When a new incident-structure is generated, it will either be merged with an existing incident structure already on the incident stack, or it will be added to the incident stack as a separate incident. When target templates are eventually generated from incident structures on the incident stack, each subevent within an incident structure will spawn its own template instantiation.

In comparing MBC with the rule-based consolidation module in UMass/MUC-3, we find that MBC tends to generate fewer spurious templates without sacrificing significant recall. However, we have seen test sets where MBC does lag behind in recall. In general, the two modules seem quite comparable in terms of overall performance, although MBC is easier to understand, maintain, and scale up. Most of the merging rules used by rule-based consolidation were incorporated into MBC, so it makes sense that the two modules exhibit similar behavior. Our decision to run MBC for MUC-4 was largely motivated by use of the All Templates metric as the official scoring metric for MUC-4. Because All Templates is maximally sensitive to all types of precision loss, it is generally advantageous to minimize spurious templates for this metric. MBC seemed better at eliminating spurious templates, so we decided to risk a possible loss of some recall for the sake of maximizing our precision.

7. But How Well Does It Work?

To balance a strictly quantitative profile of our information extraction capabilities, it is useful to examine a concrete example in some detail. Time spent on a concrete example can shed light on a variety of issues that might go otherwise unnoticed in a presentation that is restricted to scatter plots and score reports. Indeed, one of the most useful aspects of the MUC-3 and MUC-4 meetings were the "system walkthroughs." Each site was asked to present an analysis of their system operating on a specific test document, with attention directed to a set of particular questions and specific problems. Because each site was required to discuss the same text, comparisons across systems emerged very naturally without hard analysis. The system walkthroughs also painted a more intuitive picture of system capabilities and weaknesses than could be gathered from quantitative analyses. It was especially striking at MUC-4 to see that certain similarities were shared by a number of systems that posted strong performance levels.

To convey this more intuitive sense of system capabilities, we will reproduce our system walkthrough from MUC-4 here [Lehnert et al 1992b]. In this walkthrough, we trace the processing of a sample text that contains two separate bombing incidents. In general, CIRCUS generates multiple CN instantiations in response to each sentence, while memory-based consolidation (MBC) extracts information from the CNs and

organizes it within incident structures. CIRCUS and MBC work in a serial fashion: CIRCUS analyzes the entire text first, and then MBC works on the resulting concept node instantiations. But for the sake of this presentation, we will examine the effects of CIRCUS and MBC working together on a sentence-by-sentence basis.

Because our CN definitions extract information on the basis of phrase fragments, we will underline those portions of the input sentences that are important to relevant CNs. Any remaining segments of the input sentences that are not underlined are effectively ignored during semantic processing by CIRCUS. We will also show the preprocessed version of each input sentence, to indicate which items have been recognized by the phrasal lexicon (these will be catenated), and other minor transformations to the original source text. Abbreviations preceded by ">" represent punctuation marks. For example, >CO is a comma.

The first job of MBC is to partition multiple CNs into event structures which are then restructured into incident structures. As a rule, all CNs generated from a single sentence tend to fall into the same partition, so we will omit any detailed discussion of this preliminary conversion. But it is important to note that essential merging operations can take place during the creation of initial incident structures. For example, S1 illustrates how an accused perpetrator is linked to a murder because their associated CNs fall into a single partition:

S1: (SALVADORAN PRESIDENT-ELECT ALFREDO CRISTIANI CONDEMNED THE TERRORIST KILLING OF ATTORNEY GENERAL ROBERTO GARCIA ALVARADO AND ACCUSED THE FARABUNDO MARTI NATIONAL LIBERATION FRONT (FMLN) OF THE CRIME >PE)

CIRCUS triggers a murder CN from "KILLING" which picks up a victim = "ATTORNEY GENERAL ROBERTO GARCIA ALVARADO." The subject of the sentence has been recognized as such but does not enter into the murder CN. When CIRCUS encounters the verb "ACCUSED", a clause boundary is recognized. This allows CIRCUS to reset syntactic buffers and pick up "ACCUSED" as a new verb while retaining the previous subject buffer. "ACCUSED" triggers a perpetrator CN with confidence = SUSPECTED_OR_ACCUSED, accuser = "SALVADORAN PRESIDENT-ELECT ALFREDO CRISTIANI", and perpetrator = FARABUNDO_MARTI_NATIONAL_LIBERATION_FRONT. Note that the FMLN is recognized as a terrorist organization, thereby satisfying a soft constraint in the perpetrator CN. "ACCUSED" tells us to assume a less than factual confidence level within the perpetrator CN, but CIRCUS does not connect the perpetrator CN with any event description. In particular, no attempt is made by CIRCUS to resolve a referent for "the crime." The two resulting CN instantiations look like:

TYPE = MURDER
VICTIM = WS-GOVT-OFFICIAL,...noun group = (ATTORNEY GENERAL ROBERTO GARCIA ALVARADO)

TYPE = PERPETRATOR
CONFIDENCE = SUSPECTED_OR_ACCUSED_BY_AUTHORITIES
ACCUSER = WS-GOVT-OFFICIAL
    noun group = (PRESIDENT-ELECT ALFREDO CRISTIANI)
    predicates = (SALVADORAN)
PERPETRATOR = WS-ORGANIZATION,...
    noun group= (FARABUNDO_MARTI_NATIONAL
                  _LIBERATION_FRONT)

MBC's preprocessing and partitioning merge these two CNs into a single event structure before any high-level memory integration is attempted. Incident structures are designed to collapse multiple events (subevents) associated with a single perpetrator into a single structure. The incident structure for S1 looks like:

```
INCIDENT
        DATE = NIL
        LOCATION = NIL
        PERPS = (#S(PERPETRATOR
                        ID NIL
                        ORG (FARABUNDO_MARTI_NATIONAL_LIBERATION_FRONT)
                        WORD-SENSES (WS-TERRORIST WS-ORGANIZATION)
                        CONFIDENCE (SUSPECTED_OR_ACCUSED_BY_AUTHORITIES)
                        NEW-INFO NIL
                        SENTENCE 1))
        NEW = NIL
        PLURAL = NIL
        DISCOURSE-MODE = NIL
        SUBEVENT: NIL
                TARGETS: NIL  EFFECTS: NIL   INSTRUMENT: NIL
                VICTIMS: (#s(VICTIM
                                ID (ROBERTO GARCIA ALVARADO)
                                TITLE (ATTORNEY GENERAL)
                                NATIONALITY NIL
                                NUM 1
                                TYPE (WS-GOVT-OFFICIAL WS-LEGAL-OR-JUDICIAL
                                        WS-PROPER-NAME)
                                EFFECTS (DEATH)
                                SENTENCE 1))
```

Because MBC has no incident structures on its incident stack, this new incident structure is added to the stack, and the victim description is added to the victim stack.

S2: (we omit this sentence from the discussion - no alterations to memory are made)

S3: (GARCIA ALVARADO >CO &&56 >CO WAS KILLED WHEN A BOMB PLACED BY URBAN GUERRILLAS ON HIS VEHICLE EXPLODED AS IT CAME TO A HALT AT AN INTERSECTION IN DOWNTOWN SAN_SALVADOR >PE)

CIRCUS generates 5 CNs in response to this sentence. A simple CN describing a weapon is generated by "BOMB." More complicated CNs are triggered by "KILLED," "PLACED," and "EXPLODED."

The trigger "KILLED" creates a murder CN with victim = "GARCIA ALVARADO."

The trigger "PLACED" creates a location CN with instrument = "BOMB," and actor = "URBAN GUERRILLAS." This same CN also looks for a physical target inside a prepositional phrase, but it misses "ON HIS VEHICLE" because "on" is not one of the prepositions that it predicts. If the sentence had said "outside", "inside", "by", "near", "in", "under", "opposite", "across_from", or "in_front_of", instead of "on", we would have picked up this physical target. The omission of "on" was a simple oversight in an otherwise legitimate CN definition. This particular CN is specifically predicting a bomb since bombs are frequently the object of the verb "to place" in this domain.

The trigger "EXPLODED" creates a bombing CN with instrument = "A BOMB."

Note that we miss the location San Salvador in S3. Although we have a bottom-up mechanism designed to find dates and locations, it doesn't always work. All 5 CNs are placed in a single partition which generates a new incident structure containing a single subevent:

```
SUBEVENT: BOMBING
        TARGETS: NIL  EFFECTS: NIL                          INSTRUMENT: (#S(INSTRUMENT
        VICTIMS: (#S(VICTIM                                               ID (BOMB)
                        ID (GARCIA ALVARADO)                              TYPE WS-BOMB))
                        TITLE NIL
                        NATIONALITY NIL
                        NUM 1
                        TYPE (WS-GOVT-OFFICIAL WS-LEGAL-OR-JUDICIAL
                                WS-PROPER-NAME)
                        EFFECTS (DEATH)
                        SENTENCE 3))
```

When MBC receives this new incident structure, it runs a memory integration test for compatible target/victim descriptions, and determines that this new subevent is compatible with the incident structure already in memory. MBC therefore merges the two incidents, and memory acquires the fact that Alvarado was killed by a bomb.

S4-7: (we omit these sentences from the discussion - no alterations to memory are made)

S8: (VICE PRESIDENT-ELECT FRANCISCO MERINO SAID THAT WHEN THE ATTORNEY @GENERAL@S CAR STOPPED AT A LIGHT ON A STREET IN DOWNTOWN SAN SALVADOR >CO AN INDIVIDUAL PLACED A BOMB ON THE ROOF OF THE ARMORED VEHICLE >PE)

CIRCUS generates two CNs here. One fairly complicated CN is triggered by "PLACED." This CN picks up not just the bomb as a weapon, but also the individual as the responsible party, and the vehicle as a target. The second CN describes the bomb as a weapon and its link to the targeted vehicle (as before). These two CNs are largely redundant, and they are merged into a single incident structure because they share the same partition. This incident structure contains a perpetrator id = "AN INDIVIDUAL" along with the following subevent:

```
SUBEVENT: BOMBING
        TARGETS: (#S(PHYS-OBJ                               VICTIMS: NIL
                        ID (ARMORED VEHICLE)                EFFECTS: NIL
                        NUM 1                               INSTRUMENT: (#S(INSTRUMENT
                        TYPE (WS-TRANSPORT-VEHICLE)                       ID (BOMB)
                        EFFECTS NIL
                        TYPE WS-BOMB))
                        SENTENCE 8))
```

MBC checks this incident structure against the incident structure already in memory and determines that they should be merged, thereby picking up a physical target for the first time. Had we picked up this physical target from S3 as well, the target integration test would have merged the two vehicle descriptions at this point as well. Note that MBC merges the description of the perpetrator as "an individual" with the previously encountered descriptor "urban guerrillas" because the earlier description is recognized to be more specific.

S9-10: (we omit these sentences from the discussion - no alterations to memory are made)

S11: (GUERRILLAS ATTACKED @MERINO@S HOME IN SAN_SALVADOR ON APR_14_89 >CO &&5 DAYS AGO >CO WITH EXPLOSIVES >PE)

CIRCUS generates 7 highly redundant CNs in response to S11. The most comprehensive CN instantiates an attack with actor = "GUERRILLAS," target = "MERINO'S HOME," and instrument = "EXPLOSIVES." This same CN also picks up the location (San Salvador) and date (April 14) by the bottom-up attachment mechanism. Locations and dates are normally not predicted by CN definitions, but they can be inserted into available CNs via bottom-up attachment. All of this information is incorporated into a single incident structure containing a bombing subevent (an attack using explosives is understood to be a bombing). The resulting incident structure is then passed to the memory integration portion of MBC.

Just as before, MBC checks to see if the new incident can be merged into the lone incident structure currently stored in memory. But this time the new structure fails to match the existing structure because of incompatible targets. MBC cannot merge a home with a vehicle. When MBC fails to merge the new bombing incident with the old bombing incident, it moves down the target stack to see if there is another incident structure that might merge, but there are no other physical targets in memory. MBC adds the new incident to the top of the incident stack, and memory now contains two bombing incidents.

S12: (THERE WERE &&7 CHILDREN >CO INCLUDING &&4 OF THE VICE @PRESIDENT@S CHILDREN >CO IN THE HOME AT THE TIME >PE)

CIRCUS produces no output for this sentence because no CN triggers are encountered. We sometimes miss information in sentences where the only verb is a form of "to be."

S13: (A 15-YEAR-OLD NIECE OF @MERINO@S WAS INJURED >PE)

CIRCUS generates an injury CN with victim = "A 15-YEAR-OLD NIECE." This results in a subevent of unknown type with a victim id = "A 15-YEAR-OLD NIECE." When MBC receives this incident, it examines the first incident on the top of its stack to see if a merge is possible. Since no incompatible victims are found in memory for this incident (the latest bombing incident specifies no victims), a merging occurs.

S14-S17: (we omit these sentences from our discussion - no alterations are made to memory.)

S18: (RICARDO VALDIVIESO >CO PRESIDENT OF THE LEGISLATIVE ASSEMBLY AND AN ARENA LEADER >CO SAID THE FMLN AND ITS FRONT GROUPS ARE RESPONSIBLE FOR THE "IRRATIONAL VIOLENCE THAT KILLED ATTORNEY GENERAL GARCIA >DQ >PE)

CIRCUS produces a murder CN with victim = "Attorney General Garcia" and actor = "irrational violence." This CN has a soft constraint on the actor slot which specifies a human or organization, but the CN survives the CN filter because its other variable slot has a filler that does meet the required soft constraints (the filter errs on the side of spurious information if one slot looks good and the other slot looks bad). MBC is careful to check available soft constraints when it integrates information into its preliminary incident structures. Any slot fill that violates a soft constraint is discarded at that time.

When MBC attempts to integrate this incident into memory, it locates a compatible victim in the victim stack, and merges the new incident structure with the existing structure that describes Garcia as a victim. Because we have now merged new

information into an incident that was not at the top of the incident stack, we have to reorder the incident stack by moving the most recently referenced incident to the top of the stack. This effectively identifies the first incident as the current topic once again. Ideally, this would set us up to correctly integrate information contained later in S21 and S22 where new information is presented about the vehicle bombing, but CIRCUS fails to pick up the additional human targets from those sentences, so the topic shift that we've successfully recognized at S18 goes unrewarded.

When MBC completes its analysis, the two bombing incident structures are converted into two template instantiations, along with a third threat incident picked up from additional sentences near the end of the text. In order to instantiate the final templates, we rely on semantic features in our dictionary to recognize a home as a civilian residence and an armored vehicle as a transport vehicle.

We did fairly well on the first template (see Figure 8). We missed San Salvador as the location within El Salvador, we said the vehicle was destroyed instead of damaged, and we missed 3 human targets (the driver who was not hurt, and the 2 bodyguards, one of whom was injured). All the other slots were correctly filled. On the second template, we fail in three places. We have no perpetrator organization, we miss the physical target type for Merino's home (it should have been GOVERNMENT OFFICE OR RESIDENCE), and we are missing the 7 children that were human targets (this is one of the few texts where a hum-tgt-total-num slot should receive a value).

0. MESSAGE: ID	TST2-MUC4-0048	
1. MESSAGE: TEMPLATE	1	;correct
2. INCIDENT: DATE	- 19 APR 89	;correct
3. INCIDENT: LOCATION	EL SALVADOR	:partial
4. INCIDENT: TYPE	BOMBING	;correct
5. INCIDENT: STAGE OF EXEC.	ACCOMPLISHED	;correct
6. INCIDENT: INSTRUMENT ID	"BOMB"	;correct
7. INCIDENT: INSTRUMENT TYPE	BOMB: "BOMB"	;correct
8. PERP: INCIDENT CATEGORY	TERRORIST ACT	;correct
9. PERP: INDIVIDUA	"URBAN GUERRILLAS"	;correct
10: PERP: ORGANIZATION ID	"FARABUNDO MARTI NATIONAL LIBERATION FRONT"	;correct
11: PERP: ORG CONFIDENCE	SUSPECTED OR ACCUSED BY AUTHORITIES: "FARABUNDO MARTI NATIONAL LIBERATION FRONT"	;correct
12: PHYS TGT: ID	"ARMORED VEHICLE"	;correct
13: PHYS TGT: TYPE	TRANSPORT VEHICLE: "ARMORED VEHICLE"	;correct
14: PHYS TGT: NUMBER	1: "ARMORED VEHICLE"	;correct
15: PHYS TGT: FOREIGN NATION	-	;N/A
16. PHYS TGT: EFFECT	DESTROYED: "ARMORED VEHICLE"	;partial
17: PHYS TGT: TOTAL NUMBER	-	;N/A
18: HUM TGT: NAME	"ROBERTO GARCIA ALVARADO"	;correct
19: HUM TGT: DESCRIPTION	"ATTORNEY GENERAL": "ROBERTO GARCIA ALVARADO"	;correct/missing
20: HUM TGT: TYPE	GOVERNMENT OFFICIAL: "ROBERTO GARCIA ALVARADO"	;correct/missing
21: HUM TGT: NUMBER	1: "ROBERTO GARCIA ALVARADO"	;correct/missing
22. HUM TGT: FOREIGN NATION	-	;N/A
23: HUM TGT: EFFECT	DEATH: "ROBERTO GARCIA ALVARADO"	;correct/missing
24: HUM TGT: TOTAL NUMBER	-	;N/A

Figure 8: Our response template for the first bombing incident

Overall, TST2-MUC4-0048 showed the UMass/MUC-4 system working fairly well and not making any major errors. Most of our recall loss resulted from a failure to recognize relevant information in S12 (the 7 children), S21 and S22 (the driver and 2 bodyguards). As we saw in this text, the system can recover from some failures in sentence analysis when a text provides redundant descriptions (e.g. we missed the physical target in S3, but picked it up correctly in S8). When memory-based consolidation responds correctly to topic transitions, the output that CIRCUS generates usually makes it into the correct places in the response templates. TST2-MUC4-0048 shows how MBC was able to correctly recognize two topic transitions: first from an old incident to a new incident, and then back again to the earlier incident. Given that the errors encountered for TST2-MUC4-0048 were relatively minor (one could even argue that the third template was valid and should have been covered by an optional key template), there is nothing here that illustrates the more serious problems that impacted our TST3 and TST4 score reports.

Figure 9 shows score reports for the two templates that mapped to TST2-MUC4-0048 answer keys, along with the final score report for the entire message which averages in the spurious template that we generated for the threat. This final score report for the whole message illustrates how much negative impact spurious templates have on precision if a system is generating one spurious template for every two good templates. If we had generated a summary score report based on only two templates instead of three, our All Templates precision would have been 94. With the third template averaged in, our All Templates precision drops to 76.

In a domain that is characterized by complicated domain guidelines, with many grey areas, hand-coded key templates cannot be trusted to embody encodings that are necessarily superior to the output of a high performance extraction system. If this is the case, it may be very difficult to attain 85% precision under All Templates. Optimal precision levels may be closer to the 70-80% range. It has been estimated that the upper limits on human performance are around 75% for recall and 85% for precision [Sundheim 1992], but these estimates have not been substantiated by carefully designed experiments. In any case, it is clear that any task confounded by complex task specifications, limitations in the template format, and inherent ambiguities in the source texts, is not going to lend itself to perfect recall and precision levels by either humans or machines.

Vehicle Bombing Template

	POS	ACT	COR	PAR	INC	ACR	IPA	SPU	MIS	NON	REC	PRE	OVG
inc-total	6	6	5	1	0	0	0	0	0	0	92	92	0
perp-total	4	4	4	0	0	0	0	0	0	0	100	100	0
phys-tgt-total	4	4	3	1	0	0	0	0	0	2	88	88	0
hum-tgt-total	14	5	5	0	0	0	0	0	9	2	36	100	0
TOTAL	28	19	17	2	0	0	0	0	9	4	64	95	0

Home Bombing Template

	POS	ACT	COR	PAR	INC	ACR	IPA	SPU	MIS	NON	REC	PRE	OVG
inc-total	6	6	6	0	0	0	0	0	0	0	100	100	0
perp-total	4	2	2	0	0	0	0	0	2	0	50	100	0
phys-tgt-total	3	3	2	0	1	0	0	0	0	3	67	67	0
hum-tgt-total	11	4	4	0	0	0	0	0	7	2	36	100	0
TOTAL	24	15	14	0	1	0	0	0	9	5	58	93	0

Total Scores for TST2-MUC4-0048

	POS	ACT	COR	PAR	INC	ACR	IPA	SPU	MIS	NON	REC	PRE	OVG
inc-total	12	16	11	1	0	0	0	4	0	2	96	72	25
perp-total	8	7	6	0	0	0	0	1	2	3	75	86	14
phys-tgt-total	7	7	5	1	1	0	0	0	0	11	78	78	0
hum-tgt-total	25	12	9	0	0	0	0	3	16	8	36	75	25
MATCHED/ MISSING	52	34	31	2	1	0	0	0	18	9	62	94	0
MATCHED/ SPURIOUS	52	42	31	2	1	0	0	8	18	24	62	76	19
MATCHED ONLY	52	34	31	2	1	0	0	0	18	9	62	94	0
ALL TEMPLATES	52	42	31	2	1	0	0	8	18	24	62	76	19
SET FILLS ONLY	23	16	14	1	1	0	0	0	7	5	63	91	0
STRING FILLS ONLY	15	10	10	0	0	0	0	0	5	1	67	100	0

P&R	2P&R	P&2R
68.29	72.72	64.37

Figure 9: Partial and Overall Scores for TST2-MUC4-0048

8. Conclusion

It has always been difficult to evaluate natural language processing technologies. Disagreements over semantic representations and intermediate syntactic representations divide the natural language processing community and act as barriers to objective system evaluations. Additional differences over the divergent goals of various language processing systems add to the confusion. One system might focus on robust sentence analysis while another addresses the viability of a psychological memory model or perhaps a linguistic model for anaphora resolution. When there is no common ground on which to base an evaluation, we are forced to compare apples and oranges.

Given this general state of the field, there is a an understandable attraction to any natural language task that lends itself to quantitiative evaluation. Information extraction tasks have emerged as unique in this regard, but it is important to understand exactly why information extraction is playing this role when other language processing tasks are not. There is really nothing special about information extraction per se. The critical requirement for any quantitative analysis of a language processing system is the availability of precise I/O data sets. Since it is generally difficult to specify desirable I/O behavior for a machine translation system or a text-to-text summarization system, these tasks do not have I/O data sets that can be readily used for system development and evaluation. Even so, information extraction is not the *only* natural language task that can benefit from a corpus of I/O pairings. Natural language database interfaces work just as well. In fact, major research efforts with database interfaces have been grounded in quantitiative performance evaluations in much the same way that work on information extraction operates now.[10]

So it seems that the MUC effort is not unique in its ability to exploit I/O data as an evaluation tool. But there is one other aspect of MUC that is unique: the MUC evaluations have been conducted within an open conference. Any research lab in the world can apply to participate in a MUC evaluation: MUC is open to all comers and the MUC test results are openly published.

The open nature of the MUC evaluations and the public distribution of test results endows the MUC evaluations with a sense of credibility and influence that is not associated with performance evaluations restricted to a select group of contractors. Since no one is excluded from the MUC evaluations, participation in MUC confers a degree of legitimacy on those researchers who are willing to subject their claims to public scrutiny. But the primary contribution of the MUC evalutions to the field of natural language processing may be only indirectly connected to the evaluation process itself. The most remarkable contribution of MUC to the field of natural languge processing probably lies in the fact that a large number of different research groups have all examined the same task in depth for some number of months before meeting to discuss their achievements and failures. A common task facilitates communication across disparate research groups that might otherwise be inaccessible to one another. Theoretical differences, diverse educational

[10] ARPA-sponsored researchers have been working on the ATIS speech understanding project since 1989. ATIS systems are tested on their ability to interpret speech input directed at an airlines reservation system. The final test of an ATIS query system is the correctness of the responses supplied by the underlying database.

backgrounds, and paradigmatic disparities are difficult gaps to bridge, but a remarkable amount of common ground falls into place when everyone is trying to build the same system.

One can argue that a lot of the work needed to build an effective information extraction system is not of theoretical interest to computational linguists, AI researchers, or anyone else associated with academic research. This is undeniably true. Large complicated systems are not be uniformly elegant throughout. But the process of cost effective system development is an intellectually challenging problem in its own right, and a lot of the tedius work associated with building a large system might be managed in an automated or semi-automated fashion. This aspect of cost-effective software design should be embraced by the natural language community as a legitimate research concern right alongside our more traditional interests. Practical system development cannot drive theoretical research, but practical concerns can stimulate theoretical work in directions that might otherwise be ignored. If the natural language community persists in maintaining an overly narrow definition of its proper responsibilities, we will never close the gap between theory and practice in natural language processing.

In the period since our initial evaluation in MUC-3, the UMass NLP Lab has been able to pursue a number of related research projects that address basic research issues associated with CIRCUS and text extraction systems. We have demonstrated that case-base reasoning and machine learning techniques can be successfully applied to the disambiguation of relative pronouns [Cardie 1992a, 1992b, and 1992c]; experiments have shown how CIRCUS can be used to support relevancy feedback algorithms for text classification [Riloff and Lehnert 1992, Riloff 1993b]; experiments have been conducted with a statistical database derived from the MUC-3 corpus [Fisher and Riloff 1992]; we are obtaining a better understanding of the issues associated with automated dictionary construction [Riloff and Lehnert 1993]; and we have designed a portable and memory-efficient part-of-speech tagger [Lehnert and McCarthy 1993].

Our research activities since 1991 suggest that performance evaluations can stimulate basic research in compelling and beneficial ways. It is nevertheless important to maintain a proper perspective on the limitations associated with these evaluations. Score reports cannot tell us everything we might want to know about a particular system. As we saw in our discussion of automated dictionary construction, there is a distinction between viable technologies and practical technologies that cannot be readily assessed by performance evaluations alone. It is also important to reach beyond the numbers in a score report during system development in order to identify the weakest components of a complex system. When multiple components contribute to a complex behavior, there is no substitute for manual spot checks of system output. Quantitative feedback can never supplant the need for qualitative assessments.

We realize that quantitative methods lend themselves to hard analysis and are therefore methodologically clean. Hand-picked examples and casual observations provide evidence that is more properly characterized as anecdotal. But we must never underestimate the power of the right example or an incisive observation. Without a complement of intelligent interpretation, performance evaluations are necessarily limited in what they can tell us about individual systems, comparisons between systems, and the overall state-of-the-art. As useful as performance evaluations are, we must constantly remind ourselves that a meaningful assessment of a complex technology requires a level of comprehension that cannot be captured on a scatter plot.

References

Cardie, C. (1992a) "Corpus-Based Acquisition of Relative Pronoun Disambiguation Heuristics," *Proceedings of the 30th Annual Conference of the Association of Computational Linguisitcs.* pp. 216-223.

Cardie, C. (1992b) "Learning to Disambiguate Relative Pronouns," *Proceedings of the Tenth National Conference on Artificial Intelligence*. pp. 38-43.

Cardie, C. (1992c) "Using Cognitive Biases to Guide Feature Set Selection" *Proceedings, Fourteenth Annual Conference of the Cognitive Science Society.* pp. 743-748.

Cardie, C. and Lehnert, W. (1991) "A Cognitively Plausible Approach to Understanding Complex Syntax," *Proceedings of the Ninth National Conference on Artificial Intelligence*. pp. 117-124.

Chinchor, N., Hirschman, L., and Lewis, D. (1993) "Evaluating Message Understanding Systems: An Analysis of The Third Message Understanding Conference (MUC-3)," *Computational Linguistics* . 19(3):409-448.

Fisher, D. and Riloff, E. (1992) "Applying Statistical Methods to Small Corpora: Benefiting from a Limited Domain," *Probabilistic Approaches to Natural Language*, AAAI Fall Symposium. pp. 47-53.

Lehnert, W. and McCarthy, J. (1993) "A Compact p-o-s Tagger for Domain-Specific Sentence Analysis". (in preparation).

Lehnert, W. (1991) "Symbolic/Subsymbolic Sentence Analysis: Exploiting the Best of Two Worlds" in J. Pollack and J. Barnden, Eds., *Advances in Connectionist and Neural Computation Theory*. Ablex Publishing. Norwood, New Jersey pp. 135-164

Lehnert, W., Cardie, C., Fisher, D., McCarthy, J., Riloff, E., Soderland, S. (1992a) "University of Massachusetts: MUC-4 Test Results and Analysis," *Proceedings of the Fourth Message Understanding Conference*. pp. 151-158.

Lehnert, W., Cardie, C., Fisher, D., McCarthy, J., Riloff, E. Soderland, S. (1992b) "University of Massachusetts: Description of the CIRCUS System as Used for MUC-4," *Proceedings of the Fourth Message Understanding Conference*. pp. 282-288.

Lehnert, W., Cardie, C., Fisher, D., Riloff, E., Williams, R. (1991a) "University of Massachusetts: MUC-3 Test Results and Analysis," *Proceedings of the Third Message Understanding Conference*. pp. 116-119.

Lehnert, W., Cardie, C., Fisher, D., Riloff, E., Williams, R. (1991b) "University of Massachusetts: Description of the CIRCUS System as Used for MUC-3," *Proceedings of the Third Message Understanding Conference*. pp. 223-233.

Lehnert, W., Williams, R., Cardie, C, Riloff, E., and Fisher, D. (1991c) "The CIRCUS System as Used in MUC-3," Technical Report No. 91-59, Department of Computer and Information Science, University of Massachusetts.

Lehnert, W.G. and Sundheim, B. (1991) “A Performance Evaluation of Text Analysis Technologies,” *AI Magazine.* Fall 1991. pp. 81-94.

Riloff, E. (1993a) “Automatically Constructing a Dictionary for Information Extraction Tasks,” *Proceedings of the Eleventh Annual Conference on Artificial Intelligence..* pp. 811-816.

Riloff, E. (1993b) "Using Cases to Represent Context for Text Classification". To appear in *Proceedings of the Second International Conference on Information and Knowledge Management* (CIKM-93).

Riloff, E. and Lehnert, W. (1993) “Automated Dictionary Construction for Information Extraction from Text,” *Proceedings of the Ninth IEEE Conference on Artificial Intelligence for Applications*. pp. 93-99.

Riloff, E. and Lehnert, W. (1992) “Classifying Texts Using Relevancy Signatures,” *Proceedings of the Tenth National Conference on Artificial Intelligence*. pp. 329-334.

Sundheim, B. (1992) “Overview of the Fourth Message Understanding Evaluation and Conference,” *Proceedings of the Fourth Message Understanding Conference*. pp. 3-21.

Sundheim, B. (1991) *Proceedings of the Third Message Understanding Conference*. Morgan Kaufmann, San Mateo, CA.