GE NLTOOLSET: MUC-4 TEST RESULTS AND ANALYSIS

Lisa Rau, George Krupka, and Paul Jacobs
Artificial Intelligence Laboratory
GE Research and Development
Schenectady, NY 12301 USA
E-mail: rau@crd.ge.com
Phone: (518) 387 - 5059
and
Ira Sider and Lois Childs
Military and Data Systems Operation
GE Aerospace

Abstract

This paper reports on the GE NLTOOLSET *customization effort for MUC-4, and analyzes the results of the TST3 and TST4 runs.*

INTRODUCTION

We report on the GE results from the MUC-4 conference and provide an analysis of system performance. In general, MUC-4 was a very successful effort for GE. The NLTOOLSET, a suite of natural language text processing tools designed for easy application in new domains, proved its mettle, as we were quickly able to integrate the changes from the MUC-3 to the MUC-4 task.

On the positive side, MUC-4 provided a thorough, fair test of system capabilities, and allowed us to implement and test new strategies within the context of a task-driven system. Once again, the methodology of testing on a real task, along with the benefit of a common corpus, has produced advances in the field as well as highlighting certain new aspects of text interpretation. One surprise was that we continued to make improvements in sentence-level parsing and interpretation, while at the end of MUC-3 we had suspected that improvements in parsing would not yield substantial improvements to our overall performance.

On the negative side, major and significant improvements were not easy to make. Although improving the accuracy and coverage of the core language parsing mechanism accounted for some percentage of our improvements, the remainder of the gain in score is attributable to increases in the accuracy of the template post-filtering and to many small, incremental enhancements and modifications to the existing system. These "diminishing returns" continue to stand in the way of vastly improved system performance. Although there are some major problems (such as world knowledge, event-based reasoning, and reference resolution) that can be said to account for much of the remaining error in MUC, it is not clear that MUC is really measuring progress toward solving these major problems so much as progress on the many minor problems that are more easily solved.

RESULTS

Our overall results on both TST3 and TST4 were very good in relation to other systems. Figure 1 summarizes our results on these tests.

In addition to these core results, Figure 2 summarizes our performance on the adjunct test.

Finally, to put these runs in the context of our other results, Figure 3 illustrates how our system improved over time, and puts the TST3 and TST4 scores in perspective.

	TST3				TST4				
	REC	PRE	OVG	F	REC	PRE	OVG	F	
Matched Only	70	70	17		71	70	19		
All Templates	58	54	36	55.9	62	53	39	57.15	
MUC-3 Comp	50	43	46	46.2	50	43	46	46.2	
High Prec MO	60	75	8		64	77	9		(orig.)
High Prec AT	41	62	23	49.4	46	61	28	52.5	(orig.)

Figure 1: GE MUC-4 TST3 and TST4 Results

	Matched Only			All Templates			
	REC	PRE	OVG	REC	PRE	OVG	F
TST3	70	70	17	58	54	36	55.9
1MT	77	69	18	70	53	38	60.3
1ST	78	64	22	68	36	56	47.1
2MT	61	70	10	46	49	37	47.4
NST	85	78	13	43	78	13	55.4
High-prec	60	75	8	41	62	23	49.4

Figure 2: Performance on Adjunct Testing

TST3 - TST4 SCORE COMPARISON

We were very pleased that our performance on the new time slice (TST4) was virtually indistinguishable from (even higher than) our performance on the test sample from the same time as the training set. We attribute this to the fact that our system was developed and tested for general portability across subject areas, application areas, and different types of language and text. We think this is clear evidence that our approach to text interpretation is not in any way geared or slanted toward the particulars of the training set. Most of the work in the system is still done from core knowledge and basic linguistic principles.

These comparison numbers also indicate our general reluctance to encode any knowledge or write any code that was domain-specific and would interfere with general processing.

OPTIONAL HIGH-PRECISION RUN

In addition to the required testing, we performed one optional test, indicated in the HIGH-PREC rows of Figure 1. Our system could not produce significantly higher recall without effectively guessing, so we decided only to reconfigure the system to produce a high-precision result. First, we noticed that most of our errors were being introduced through the incorrect application of template merging decisions. There are fewer of these decisions when all the information about an event appears in a sentence. Also, our single sentence level

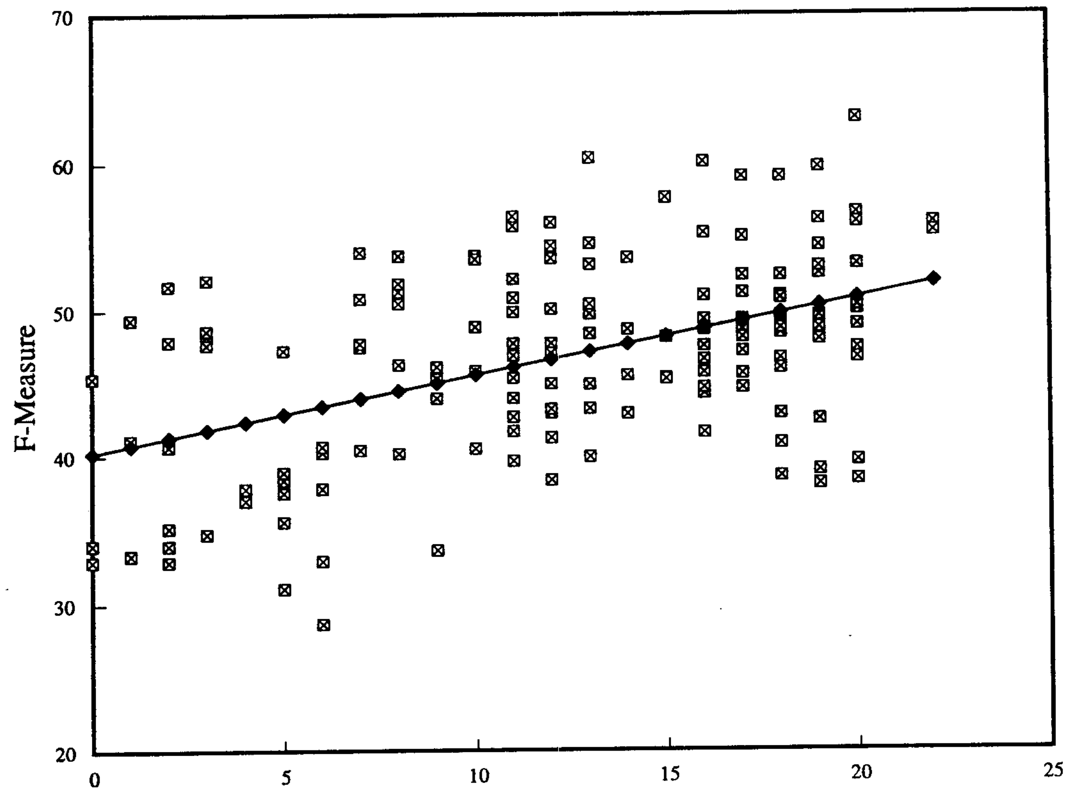

Figure 3: Improvement

processing was more accurate than our multi-sentence processing, so this strategy was likely to produce high precision.

For the high-precision configuration, we set the system to use only one sentence to fill the content of each template, using the single sentence for each template that contained the most fills. This strategy is very crude, and more clever methods are likely to improve upon this. For example. we could perform selective merging of information from multiple sentences when there is a high degree of overlap between the two. However, even this crude method produced significantly higher precision (15% higher on TST3 and 22% higher on TST4).

EFFORT

We spent overall about 10 1/2 person-months on MUC-4, as compared with about 15 person-months on MUC-3. This time was divided as follows:

2 mo: Knowledge Additions: New or altered patterns, grammar rules, activators, domain expectations, names, and places. Complete addition of all possible target fills. Addition of primary and support templates, lexicon, phrases, patterns and hierarchy.

1 mo: New Place and Time Mechanism: A new and clean location and time handler was integrated into the Toolset.

1 mo: Answer Key Mechanism: Design and implementation of mechanism to use information extracted from a canonical, conceptual version of the entire answer key.

2 mo: Parser Improvements: Parser recovery and improving attachment.

1 mo: TRUMPET upgrade: Revision of domain expectation mechanism to use structure sharing.

1/2 mo: MUC-4 Upgrade: Upgrade of MUC-specific mechanisms to be compatible with new MUC-4 format.

Type	Missing	Spurious
Post-processing filter; relevancy	18%	57%
Bugs, glitches	23%	17%
Code-level	29%	20%
Knowledge-level	30%	6%

Figure 4: Attribution of Error in TST3

1 mo: Misc. Bug Fixing: Hundreds of small bugs were found and fixed.

2 mo: Scoring, Reporting: Meetings, reporting, incremental and final scoring, analysis and other overhead.

We performed an in-depth study to attribute all errors in the TST3 run to components of the system. For TST3, we had a total of 24 missing templates and 33 spurious templates. For TST4, we had 13 missing templates and 33 spurious templates. Moreover, **76%** of the spurious points came from whole spurious templates, whereas **41%** of the missing points came from missing templates. This indicates the the largest single source of immediate improvement in our score should come from increasing the accuracy of our template filtering stage. Template filtering is the process when we determine after a template has been filled out if it is spurious due to relevancy conditions.

The Figure 4 summarizes the source of error in terms of the percentage of points between our score and a perfect score.

Most of the code errors were due to inaccuracies in the determination of event boundaries; part of the "discourse module". In fact, **25%** of our missing points come from inaccurate reference resolution, and event splitting and merging problems. We hope to address these problems in the next improvements to the NLTOOLSET.

TRAINING

Our method of training was to run our system over the messages in the development and TST1 corpus. We kept the TST2 set of messages and answers separate as a safeguard to over-training. We used the results of these runs to detect problems and determine where we needed additional effort.

We experimented with using the answer key as an aid in the automatic acquisition of domain-specific knowledge. In particular, the entire development answer key was canonicalized by transforming all natural language strings present as fillers of slots in the key to their conceptual heads. Second, generalizations were extracted to reflect reliable information on the habitual roles certain concepts play in the database domain of the texts. This process was found to be useful in four places:

Detecting Gaps in Knowledge: By sending all the strings present in the answer key through our natural language system, we can detect errors, gaps in knowledge and other problems within the domain of the answer key. This process is a prerequisite to using the answer key, as it produces a canonical, conceptual version.

Determining Valid Generalizations: Certain combinations of fills always occur together. These generalizations are automatically detected and used to prevent incorrect slot filling. For example, the terrorist organization **SHINING PATH** always carries out its terrorist activities in **PERU**.

Determining Hard Constraints: Certain fillers make particularly good fillers for certain slots. For example, someone described in a text as a **VICTIM** is a much better filler for the **TARGET** slot than the **PERPETRATOR**. These constraints serve to prevent inappropriate fillers from appearing.

Encoding Specific, Recurring Events: In certain domains, and with certain types of texts, frequently recurring events can be encoded more specifically to aid in the accuracy of their interpretation.

Anomaly Detection: Finally, a canonical answer key, when compiled into lists of unique fillers for each slot, allows for the easy detection of incorrect answers.

Our initial test runs on the MUC-4 TST3 and TST4 showed a small increase (1 point each) in both recall and precision on TST3 and a negligible effect on TST4 from the answer key training. In particular, the combined (recall and precision) measure was 53.93 without these data, and 54.90 with the data for the TST3 (same time period as the answer key) test. Recall went from 56 to 57 and precision went from 52 to 53. Although these increments may seem small, our experience has been that any noticeable increase in performance is significant at these levels of accuracy. That is, as systems make fewer and fewer mistakes in interpreting texts, it becomes more and more difficult to find areas for any improvements.

For the TST4 time slice, as we anticipated, there was no noticeable effect of answer key training. Aside from the use of the knowledge to fill gaps in the system's knowledge base, the information in a novel set of messages does not intersect with the information extracted from an old set of messages. Thus, the answer key training neither helps nor hurts novel messages.

In addition to this experiment, which used training based on an answer key to resolve template-level decisions, we tried several methods for corpus-based training to help with sentence-level interpretation. These experiments included corpus-based part-of-speech tagging, statistically-based information to help with attachment, and a "last ditch" method for guessing a parse where the parser produced a suspicious result. In some cases, these methods showed a marginal improvement in early tests. However, as we neared the final MUC test, none of them showed any positive effect. We treat this as evidence that it is difficult to use automated training to guide sentence-level interpretation when sentence-level accuracy is already very high.

RETROSPECTIVE ON THE TASK AND RESULTS

In retrospect, we probably could have made additional improvements to performance if we had made significant changes to the mechanism that splits stories into events, and the mechanism that resolves references (including definite anaphora and multiple descriptions of the same object or event). Aside from these areas, all the other portions of the NLToolset are working quite well.

The speed of our system, around 500/words per minute on this task, understates its real speed due to a non-optimized configuration. Nonetheless, this speed is achieved on conventional hardware and is already way ahead of human performance. This suggests that this technology will be able to process large volumes of text. We were able to process TST3 in 1 hour and 28 minutes, and TST4 in 1 hour and 5 minutes.

We were similarly pleased that the sentence-level performance of the NLToolset was as good as it was. While we fixed minor problems with the lexicon, grammar, parser, and semantic interpreter, robustness of linguistic processing did not seem to be a major problem. For MUC-3, we believed this was because because the domain was still quite narrow. It was much broader than MUCK-II, and the linguistic complexity is a challenge, but knowledge base and control issues are relatively minor because there are simply not that many different ways that bombings, murders, and kidnappings occur. However, given the improvements we made to the sentence-level interpretation in MUC-4, we feel that the correct interpretation of individual sentences can have a significant effect on the overall accuracy of interpretation. This is partly due to our observation that the effects of parsing "fan out" to affect the accuracy of other components of the system. Also, we changed other components to take advantage of the increased accuracy of the interpretations.

REUSABILITY

We estimate that about 70% of the knowledge encoded for this effort is reusable, whereas over 80% of the code is reusable. This includes most of the improvements to the parser recovery (or 20% of the total effort).

LESSONS LEARNED

The GE System

This task has proven our system's transportability, robustness and accuracy. The major sources of improved performance were increasing the accuracy of the template filtering, the coverage and robustness of the parser, and error recovery handler. The other improvements have come from the additive effect of many little enhancements and fixes throughout the system (See the GE system summary paper in this volume).

There is a bit of a puzzle in the observation that our improvement seems to have come from doing better at things we already could do well, while the error that remains seems to come predominately from problems we don't really have solutions for at all, like general reference resolution and reasoning about background information. One conclusion that we have drawn from this analysis is that our progress on these major problems will come not from new modules, but from adding new sources of knowledge to our existing modules. This was not the approach we had taken in MUC-3.

Our experience with MUC-4 has pointed out the need for much closer integration of the discourse (event and reference resolution) components of the system with the control and core language understanding components. It is clear to us that increased performance will not come easily from separable modules.

These will continue to be the areas where we hope our system will improve over time.

Evaluation and the MUC Task

We continue to believe that MUC is an interesting, realistic task and that it advances the state of the art in natural language processing as well as providing a good test of system capabilities.

The move from MUC-3 to MUC-4 produced some clear improvements in the test as well as the systems. For example, the emphasis on ALL TEMPLATES was clearly a better way of evaluating systems, which none of the sites seemed to recognize prior to MUC-3. The minimal matching constraints prevented some of the rewards for overgeneration that were a problem in MUC-3. The combined F-measure, while hiding many of the interesting aspects of performance, at least gives an explicit basis for comparison.

It would be nice if the price tag for these tests could be reduced, by reducing the time and effort required to run through the mechanics of executing and evaluating each test, and perhaps making formal evaluations less frequent. In addition, a new domain with a smaller amount of time for development might be more rewarding than repeatedly testing on the same task and domain. New tasks or domains with short preparation times: (1) minimize the work that each site can do that is task-specific, (2) allow new systems to participate on equal footing with others, and (3) test transportability and general techniques by preventing too much specific development or knowledge coding. The strategy of coming up with new domains could make it harder to show the overall progress of the field, but it is actually at least as hard to attribute progress in the MUC-3 - MUC-4 sequence as it was in the MUCK-II - MUC-3 sequence, which represented a much more drastic shift in task and domain.

One of the most promising new areas that has emerged from these evaluations is the ability, within a given system, to test new modules, strategies and configurations, as a controlled way of testing the impact of a particular algorithm or strategy. We have learned not only from our own experiments of this sort, such as the comparison between the GE and CMU parsers (see the GE-CMU site report and system summary in this volume), the high-precision run, and the answer key experiment, but also from the controlled experiments that *other* teams have done (such as, in previous MUC's, NYU's analysis of recovery strategies and SRI's decomposition of errors). This whole style of experimentation should be embraced as one of the most rewarding, non-competitive aspects of the evaluations.

SUMMARY

The GE system performed well, as we had hoped, on both TST3 and TST4. The tests proved some of the positive advances we had achieved, as well as surprising us somewhat by showing progress in areas where our system was already strong. Some of the experiments and analyses we did as part of the test were as rewarding as the comparative results. The whole MUC experiment has thus opened up a new methodology that we are beginning to explore in using comparative and controlled testing to guide algorithmic research.