GE NLTOOLSET: MUC-3 TEST RESULTS AND ANALYSIS

George Krupka, Lucja Iwańska, Paul Jacobs and Lisa Rau
Artificial Intelligence Laboratory
GE Research and Development
Schenectady, NY 12301 USA
rau@crd.ge.com

Abstract

This paper reports on the GE NLToolset *customization effort for MUC-3, and analyzes the results of the TST2 run. Although our own tests had shown steady improvement between TST1 and TST2, our official scores on TST2 were lower than on TST1. The analysis of this unexpected result explains some of the details of the MUC-3 test, and we propose ways of looking at the scores to distinguish different aspects of system performance.*

INTRODUCTION

We report on the GE results from the MUC-3 conference and provide an analysis of system performance. In general, MUC-3 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 show good results on the MUC-3 task. Even on TST2, where we experienced some system-level problems, all of our results were in the top group, and the program was especially accurate at filling out templates. There were, however, some surprises that resulted from MUC-3, including the major differences in system capabilities that are largely hidden in the scores, as well as the relative ease of sentence-level interpretation.

On the positive side, MUC-3 provided a thorough, fair test of system capabilities. 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. Certain parts of our system, including a lexically-driven pre-processor and knowledge-based language analysis mechanism, worked extremely well, while other issues, such as our lack of an explicit discourse representation, prevented us from doing better. This recognition of strengths and weaknesses comes directly from the results of the MUC-3 experiment.

On the negative side, too much of each system's score, as well as the work involved in the task, is from applying "rules of the game", and future MUCs must try to minimize this component. Some of these rules are not tied either to text processing capabilities or to the practical requirements of the task. Another problem is how to determine from the results what a system is actually doing, as some major differences between systems proved largely hidden in the MUC-3 scoring. In this report, we will attempt to relate the evaluation results to system capabilities as well as to suggest some methods for attributing different aspects of the scores to particular types of processing.

RESULTS

Our overall results on TST2 were very good in relation to other systems, but we devoted much of our analysis to explaining why they were much lower than our expectations.

The GE results on TST2 are unusual in that we experienced a considerable *drop* in performance between TST1 and TST2, in spite of enhancements to our system that showed substantial improvement on our testing prior to TST2. Part of the drop is attributable to system-level problems introduced directly before the test. To determine the effect of these problems, we produced a revised run with two one-line changes in the system code. However, even this revised run shows a significant difference between runs on TST1 and the development corpus and the TST2.

The following table summarizes our results on the second test, TST2, both officially and with the system problems corrected.

	Revised Run			Original Run		
	REC	PREC	OVERGEN	REC	PREC	OVERGEN
Matched Only	62	62	20	58	63	18
Matched / Missing	52	62	20	42	63	18
All Templates	52	45	42	42	46	40
Set Fills Only	50	61	22	39	60	21

Figure 1: GE MUC-3 TST2 Revised Results* (Unofficial) vs. Official

In addition to these core results, we ran a number of other tests to put the TST2 runs in the context of our other results. Figure 2 illustrates how our two runs TST2 (official) and TST2* (revised) compare with the historical system performance on training data. Data points that share an X coordinate represent runs using the same system configuration on distinct 100-message samples taken from the development corpus.

Although the TST2* (revised) point is clearly more representative of system performance than TST2, we were still surprised by the drop and did some analysis to try to determine its cause. While we cannot definitively explain why the TST2 points are lower, the lower performance on TST2 *does not* seem to indicate that our system was overly tuned to the development examples. To test this, we restored the system from tape to a configuration as close as possible to the TST1 run. This point, marked on the Figure 2 graph as TST2 in March, is still about 10 points lower in recall than the TST1 run. In addition, note that the range of recall scores on different sets of 100 texts from the development corpus, shown by unlabeled dots at any fixed time on the graph, is about 20 points, a substantial variation.

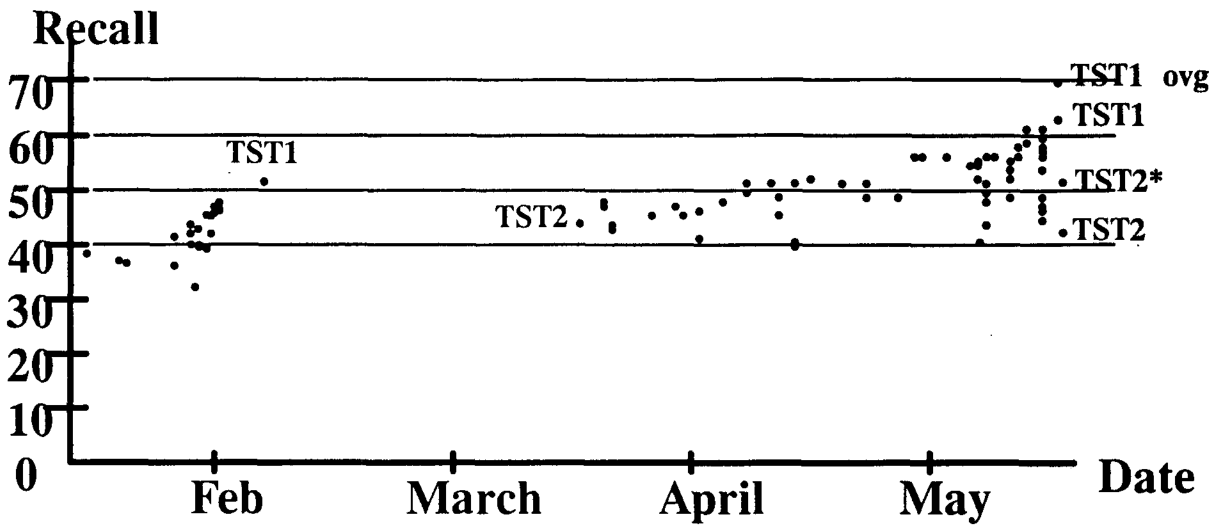

Figure 2: Improvement

Site	M/M REC	M–O REC	Templates	Spurious	Template Ovg
UMass	51	54	215	108	50%
NYU	44	49	187	90	48%
BBN	45	45	296	179	60%
GE	42	58	105	31	30%
HUGHES	31	41	163	77	47%
PRC	28	32	174	68	39%
MCDAC	28	39	118	42	36%
SRI	25	44	83	27	32%
ITP	20	35	79	21	26%
UNISYS	19	47	47	9	19%
LSI	16	32	81	29	36%
UNEB	14	22	143	81	57%
ADS	12	14	221	124	56%
GTE	11	28	84	44	52%
SYNCH	7	18	63	31	49%

Figure 3: Template Overgeneration and Recall (Official TST2 Run)

Although the much lower performance on TST2 could fall within the normal variation of performance among different message sets, we are still left to explain this variation, which did not seem to hit other systems as hard. The most likely hypothesis is that our program performed substantially lower on TST2 than on other runs, because the strategy we chose in the final configuration was overly cautious in producing templates, while the answer key had an unusually large number of templates. This hypothesis is supported by the higher performance of our system in the MATCHED-ONLY row (see Figure 3 below). The fact that our program produced less than half as many templates as the system with the highest MATCHED/MISSING recall, combined with the fact that the answer key contained more templates than other sets, adds to the evidence that our program paid a "recall penalty" for generating fewer templates.

To test this theory, we conducted a number of experiments, two of which involved using different strategies that we had viewed as being sub-optimal. In one test, we eliminated all portions of code that cut out spurious templates, causing the program to generate about twice as many templates per message set, where most of the additional templates were incorrect (because the code had been specifically designed to eliminate *incorrect* templates, not *correct* ones). This change, certainly not one that improved the program, resulted in a 6-point gain in recall on the TST1 set (shown by TST1 ovg in Figure 2) with a 3-point loss in precision.

Then, we tried an experiment by blindly copying every template in our answer key (without changing the program or the answers otherwise). This resulted in a 6-point gain in recall with a 6-point loss in precision. Since these extra templates could not possibly be matching correctly (because no two events should be alike), this experiment also shows that generating incorrect templates tends to result in higher recall than not generating enough templates, and suggests that overgenerating more intelligently tends to improve recall more than it hurts precision.

The TST2 set contained far more templates, as well as far more optional templates, than TST1 or the average for 100 messages in the development set. The development answer key contained, on average, 8 optional templates per 100 messages, while TST1 had 7 and TST2 had 32. The development answers averaged 83 filled templates per 100 messages, and TST1 and TST2 had 95 and 130, respectively.

Systems that overgenerate at the template level tend to be more impervious to changes in the percentage of OPTIONAL templates because extra templates are more likely to match, perhaps felicitously. In addition, overgenerating at the template level helps to prevent missing non-optional templates, which have the greatest effect on MATCHED/MISSING recall.

Figure 3 gives a concise summary of the number of templates each system generated with respect to their recall in MATCHED/MISSING (M/M REC) and MATCHED-ONLY (M-O REC). Our system kept its template overgeneration very low. In fact, only 3 sites had lower template overgeneration, none of them within 20 recall points. One system with slightly higher recall produced 148 additional spurious templates. Note that the systems with lower template overgeneration also tend to get a bigger gain in recall in MATCHED-ONLY.

The results seem to show a surprising variation from one test set to another, as well as an important tradeoff between template overgeneration and recall, especially in the important MATCHED/MISSING column. In retrospect, we believe that our overall TST2 results would have been closer to the expected performance of our system had we been less cautious about avoiding spurious templates. On the other hand, it might have been a good idea to measure template overgeneration (as well as the "accidental" matching of templates) as part of the results, since these incorrect templates are not a good thing. In most systems, overgeneration probably came from trying to maximize MATCHED/MISSING recall, so the MUC-3 score reporting didn't suggest that template overgeneration was a real issue. Probably the test design for MUC-4 should show the relationship between template performance and overall scores more clearly.

EFFORT

We spent overall approximately between 1 and 1.5 person-years on MUC-3. This time was divided as follows:

1 mo: Pre-Processing: Pattern acquisition, prepositional phrase handling, and handling of lists of people, targets, locations, etc.

1 mo: Semantic Interpreter Improvements: Reference resolution mechanism.

4 mo: Discourse Processing: Design and implementation of mechanism to determine portions of text that describe different events.

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

1 mo: Knowledge Engineering: Addition of primary and support templates, lexicon, phrases, patterns and hierarchy.

1 mo: MUC-specific Engineering: Implementation of target typing, location handling.

5 mo: Misc. Bug Fixing: Isolation of problems and fixing.

1 mo: "Improvements": Template merging, template thresholds.

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

Primary Limiting Factor

The primary limiting factor in performance on MUC-3 was the limited ability of programs to perform linguistic and extra-linguistic tasks at a pragmatic or discourse level. These tasks include event reference resolution and inference. For example, some correct templates in TST2 depended on distinguishing two events based on the knowledge that Cartagena is a resort, assuming that two men leaving a package in a restaurant could be planting a bomb, and generating an extra template for a series of kidnappings because one of them took place on a particular day. These many discourse and event-based issues overwhelm the relatively minor problems of parsing and semantic interpretation. Robustness of linguistic processing for MUC-3 was surprisingly easy to achieve, while the intricacies of template generation were surprisingly difficult to master.

Training

Our method of training was to run our system over the messages in the development and TST1 corpus. We used the results of these runs to detect problems and determine what new capabilities we needed to make these test stories work. We did not perform any automated training, although we did make heavy use of a keyword-in-context browser and some use of data from a tagged corpus.

As explained above, lexical coverage and parsing *did not* seem to stand in the way of major performance gains for MUC-3, so we did not focus our efforts in these areas.

Our system improved fairly steadily over time, as the graph shown in Figure 2 illustrates.

These improvements were gained through a combination of adding knowledge, fixing bugs, adding some capabilities (like template splitting and merging) and coding MUC-specific tasks (like distinguishing guerrilla warfare from terrorist activity).

RETROSPECTIVE ON THE TASK AND RESULTS

In retrospect, over the last six month period, there were no major changes to our system that we would have made for MUC as a result of our experience with this corpus and task.

With a minimum of customization (perhaps one or two person months of effort), our system quickly reached the level of performance on MUC-3 achieved by the other top systems. This ultimately proved a bit discouraging, as progress from that point on was quite slow, but it is evidence that the NLTOOLSET system, designed for easy adaptation to new tasks and domains, does what it is supposed to do.

The most successful portion of our system that was designed for this task was the text reduction mechanism [1]. The NLTOOLSET now uses a lexico-semantic pattern matcher as a text pre-preprocessor to reduce the complexity of the sentences passed to the parser. This allowed us to keep the system running in real time, prevented the parser from dealing with overly complex sentences, and achieved more accurate results. In addition, the pre-processor allowed a discourse processing module to divide the input text roughly into events prior to parsing, which seemed to have a considerable positive effect on later processing (see the paper on discourse in this volume)

The speed of our system, over 1000/words per minute on this task on conventional hardware without any major optimizations, is already way ahead of human performance and suggests that this technology will be able to process large volumes of text.

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. In part, this seems to be because the MUC-3 domain is still quite narrow. It is 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.

The fact that sentence-level interpretation wasn't a major barrier in MUC-3 has both good and bad implications. Fortunately, we can expect that progress in new (perhaps extra-linguistic) areas will soon bring system performance on this sort of task ahead of human performance, and make this research pay off in real applications. Unfortunately, it is unclear whether this new progress will spill over into other domains and applications, or whether it will lead to narrowly-focused development for future MUCs. The combination of a narrow domain with broad linguistic issues could make non-linguistic solutions more attractive for this sort of task. The only way to test the degree to which these solutions are reusable is to keep testing system transportability and evaluating performance on new and broader tasks.

ISSUES IN EVALUATION

After analyzing the results of our system and the primary measures of comparison between systems (recall, precision and overgeneration in the MATCHED/MISSING row), we realized that several factors in system performance were being confounded and/or not being measured. We isolated six, interrelated measures of system performance as follows:

1. **Recall:** Gross, overall recall can be estimated by the MATCHED/MISSING column.

2. **Precision:** Gross, overall precision can be estimated by the MATCHED/MISSING column.
3. **Template Overgeneration:** The OVERGENERATION column in the ALL-TEMPLATES score report is the best overall measure of template overgeneration.
4. **Slot Overgeneration:** Subtracting the TEMPLATE-ID scores from the MATCHED/MISSING overgeneration column results in slot overgeneration.
5. **Quality of Fills:** Recall and precision in the MATCHED-ONLY row, when template-ID and spurious templates are subtracted, provides an approximation to how well the templates that are filled out are filled out.
6. **Template Match:** There are two aspects to how well a system matches the templates that are in the answer key. One is the number of templates systems generate, and the other is how accurate the types of those templates are. The precision of the TEMPLATE-ID row gives a measure of how close the number of answer templates were given. Precision and recall of the INCIDENT-TYPE slot also give the accuracy of the templates matched.

Any measure of the performance of a data extraction system must have a meaningful way of combining the effects of template level decisions with slot-filling ability, but must also distinguish slot-filling from template decisions for system comparison. Template-level decisions are:

- When to create a template
- What type of template to create
- When to merge multiple templates
- When to eliminate a template

Template-level decisions reflect a system's ability to carve out messages into discrete topics or individual events. This includes text-level issues such as when a new event is being introduced as opposed to giving further detail on an already mentioned event, and determining the topic or type of that event.

Slot-level decisions relate to the quality of the template fills once the decision has been made as to which and how many templates to generate. In general, slot-level decisions are closer to and represent more the core language processing capabilities than template-level decisions.

The interaction of recall, precision and overgeneration presents additional challenges in evaluating systems, and MUC-3 should provide ample data to test the utility of combined metrics. In addition, it is important to be able use the scores on the MUC task both for comparing systems and for proving the ultimate utility of the systems. The MUC-3 results might seem low to those not really familiar with the tests, while many of the systems could already be extremely useful even without major improvements in performance.

Finally, estimates providing a margin of error for all the scores on a MUC-like task are necessary in order to compare results meaningfully. This error comes from the inherent imprecision in any "right answer" against which scores are computed, and the inevitable difference in the performance of systems from one test set to another.

LINGUISTIC PHENOMENA TEST

Our results on the linguistic test of apposition are interesting, as we estimate that we recognize 90% of these syntactic structures with regular expression patterns in a context-independent pre-processing stage, prior to the application of any syntactic parsing using our context-free grammar.

The slot configuration files confounded the pure test of recall and precision with respect to apposition by not factoring out entire templates that were missed (presumably an issue not related to the treatment of the appositive). Also complicating a "pure" test is the penalty for spurious fills included in those slots where the appositives were present; again, an error unrelated to the fill that contained the appositive.

We corrected for these interfering effects to get a truer measure of the performance. This was done by eliminating from the test score those slots not present because of missing templates, and eliminating the spurious slot fills. With these corrections, we calculated recall for the "easy" cases to be 96% from 72% (unrevised). The hard cases went from 43% recall to 89% recall (again, unrevised). This difference is entirely attributed to one example. Based on this, we would not want to draw any substantive conclusions on our performance of easy vs. hard appositives.

Our results on the linguistic phenomena tests show that our performance on the same sentence appositives was better than the same information distributed across multiple sentences. This was expected, as our system does not use the semantic interpretation of "to be" sentences to modify the type assignments of targets. The cases here where the assignments were correct were cases of our default typing, CIVILIAN.

The preposed appositives were more accurate than the postposed. We would have expected that postposed would be easier because it is easier to determine their boundaries. Preposed appositives, on the other hand, are typically shorter and do not appear next to or in list constructs.

We would not want to draw any conclusions from these results on the intrinsic power of the pertinent techniques. These techniques are detailed in the system walkthrough paper (cf. this volume). We feel that a fair amount of effort has gone into system development for the apposition, so, from this regard, these tests seem to reflect that linguistic phenomena are not as important for overall performance as other factors. That is, larger gains in terms of recall and precision scores seem to come with less effort from focusing on discourse and event structure rather than local linguistic issues such as apposition.

REUSABILITY

We estimate that about 50% of the effort spent on this task will not be reusable at all (except, perhaps, for future MUCs), although 80% of the improvements to the parser recovery (or 20% of the total effort) are reusable. Note, however, that these are not necessarily the changes we would have chosen to make! About 10-15% of the total effort is work that is necessary for any template generation task from text in a new domain. The other 35-40% of the non-reusable effort stems from MUC-3 specific rules not tied to the effort of data extraction in general or in particular. The items that went into this effort are discussed more in Section below.

LESSONS LEARNED

The GE System

This task has proven our system's transportability, robustness and accuracy quite well. The things that worked particularly well for MUC-3 were:

```
pattern matching pre-processor
discourse processing
lexicon
parser
semantic interpreter
partial parser
```

The MUC experience also pointed out some clear deficits with some aspects of text-level interpretation that are particularly critical in multi-template texts, in particular:

```
discourse and complex event representation
reference resolution
handling background events
```

In addition, there were three problems with our system that were largely fixed during MUC-3:

```
list processing (including coordination)
phrase attachment and parser control
```

The MUC Task

Certain aspects of this MUC task did not test the text processing capabilities of the systems. These fall into the category of task-specific rules to eliminate correctly filled-out templates. The application of these rules is outside the language processing components of the systems; however, the misapplication of the rules can have a great effect on the score. We estimate three-quarters of our missing templates and most of the spurious templates are due to the misapplication of the following "rules", further described below— *stale data, guerrilla warfare, non-specific events,* and *template splitting*

We estimate that these specific problems account for approximately 50% of the missing recall in our results (i.e. half of the difference between our recall and 100% recall). The rest of the missing recall is a combination of sharing information across templates, language analysis failures, knowledge failures, and subtle differences in interpretating events. Looking at recall, this is supported by our score on the MATCHED-ONLY row, which is an underestimate because it still includes many problems in incorrectly splitting or merging templates.

The four major MUC-specific issues are:

Stale Date: Eliminate all templates that report on events over two months old, unless they add new information. The application of this rule depends on correctly determining the date of the event; an error in this slot will cause the incorrect deletion of the entire template, while extra templates and slots can result from missing the "stale date".

Guerrilla Warfare: Eliminate all templates that report on guerrilla warfare events as opposed to terrorist events.

Non-specific Events: Eliminate all templates that report in a non-specific events.

Template Splitting: Deciding on when to generate a separate template based on the granularity of the reported locations and dates for any given incident.

We believe that, to test text processing systems, fine lines of distinction between relevant and irrelevant texts should be left to human beings, and that the MUC task should focus on accurate information extraction, not subtle judgements of relevance or validity. One proposal, which has been tentatively adopted for MUC-4, is to encode these distinctions as slot fills as opposed to template/no-template decisions; for example, GUERRILLA-WARFARE could be a TYPE-OF-INCIDENT as opposed to an IRRELEVANT template. This will minimize the influence of the extra-linguistic post-editing and maximize the testing of the core system ability to extract information from text.

Evaluation

The most important lesson we learned on this task, and probably the biggest contribution of MUC to the state of the art, is the importance of having an "answer key" to direct the focus of research efforts. Without the answer key, we would proceed by fixing problems with our system, sentence by sentence. This methodology succeeds in making particular sentences and texts work, and can also fix general problems with the system. However, concentrating on sentences and phenomena, rather than tasks and answers, can also introduce unintended effects, and can focus research on phenomena that prove irrelevant to a task.

The answer key allows system developers to focus attention on fixing widespread problems as well as quickly testing the global effect of every change.

Another important lesson from this evaluation is that drastically different techniques could produce similar answers, while many important differences between systems are "buried" in the more detailed reports of scores. This happened because MUC-3 really combined many different tasks, from template generation and slot filling to temporal interpretation, knowledge-base issues, and even event recognition (e.g. knowing that Jesuits are a good target). One of the challenges for this sort of evaluation is to determine not only what produces good overall results, but also which portions of the task are best covered by which technologies.

THOUGHTS FOR MUC-4

Two competing designs for future MUCs are to retain the same domain, perhaps deepening the task, and to move on to a new domain with the same basic template-filling task. Retaining the same basic domain and task has the apparent advantage of minimizing the effort required just to perform the test, at least for those groups that have already invested the effort. The stable task also allows MUC to be used as a benchmark for measuring the progress of the field. On the other hand, keeping the task and domain stable could put new groups (i.e. those not involved in MUC-3) at a disadvantage, and runs the risk of having effort unknowingly devoted to MUC-specific problems.

The alternative, to select new tasks and broader domains for future MUCs, has the benefit of allowing new projects to enter on a roughly equal basis, to check the validity of the MUC-3 task, and to measure transportability across domains. However, this choice would require additional work of all participants, and would probably require holding the evaluations less frequently.

Presently, it seems that MUC-4 will follow the line of MUC-3, measuring the progress of the field (and the individual participants) but not showing the relationships between domains or transportability, and not introducing new capabilities. The field is moving quickly enough, however, that broader domains and new tasks will soon be necessary to have better measures of problems, progress, and applications.

Another major issue in MUCs is how often they should occur. We believe that it is far more dangerous to have the tests too frequently than to have them infrequently. While infrequent tests produce less data and provide less of a chance for new entrants, frequent evaluations of this sort are more likely to inhibit research by pushing short-term system issues in front of larger, critical advances. Perhaps the best compromise is to have continual evaluations, but expect that each site will participate only once in every two or three evaluations.

We believe that MUCs can only be a useful test of text interpretation technology if they measure transportability and customizability as well as accuracy. Otherwise, it will not be clear how much functionality is produced by special-purpose features. This could be achieved by moving to a new domain and shortening the length of the development time. Also, the task should minimize or eliminate domain-specific rules that move systems away from their information extraction role. This will give truer measures of a text processing system's ability to move into a new domain and extract useful factual information from free text.

SUMMARY

The GE system performed very well on MUC-3, but our official run on TST2 produced scores substantially lower than our TST1 results, in spite of other tests that showed system improvement over time. Even in a revised run that fixed system-level problems, our TST2 score was about the same as on TST1. In trying to explain why the performance was lower than our expectations, we made some interesting observations about the test, including the apparent relationship between template overgeneration and recall. The result of this analysis is that while the highest scoring systems all produced comparable results in the MATCHED/MISSING row, there are major differences in the way the systems produced the results. We propose several ways of finding these differences in the score reports, as well as one correction to the test design to reduce some problems with template-level decisions. Finally, we strongly support the methodology of MUC while warning against repeated, prolonged testing in any single domain.

References

[1] Paul S. Jacobs, George R. Krupka, and Lisa F. Rau. Lexico-semantic pattern matching as a companion to parsing in text understanding. In *Fourth DARPA Speech and Natural Language Workshop*, San Mateo, CA, February 1991. Morgan-Kaufmann.