SRI INTERNATIONAL FASTUS SYSTEM MUC-6 TEST RESULTS AND ANALYSIS

Douglas E. Appelt, Jerry R. Hobbs, John Bear, David Israel, Megumi Kameyama, Andy Kehler, David Martin, Karen Myers, Mabry Tyson

SRI International
Menlo Park, California 94025
appelt@ai.sri.com
(415) 859-6150

INTRODUCTION

SRI International participated in the MUC-6 evaluation using the latest version of SRI's FASTUS system [1]. The FASTUS system was originally developed for participation in the MUC-4 evaluation [3] in 1992, and the performance of FASTUS in MUC-4 helped demonstrate the viability of finite state technologies in constrained natural-language understanding tasks. The system has undergone significant revision since MUC-4, and it is safe to say that the current system does not share a single line of code with the original. The fundamental ideas behind FASTUS, however, are retained in the current system: an architecture consisting of cascaded finite state transducers, each providing an additional level of analysis of the input, together with merging of the final results.

This paper will describe the version of the FASTUS system employed in MUC-6 and highlight the innovations that distinguish it from previous versions described in the literature.

SRI used the FASTUS system for each of the MUC-6 tasks: the named entity task, the template-entity task, the coreference task, and the scenario template task. Because a single system, with a single configuration, was used to run all the tasks, and because the first three tasks are in some sense prerequisites to the fourth, we will focus our attention in this paper on the scenario template task.

BASIC FASTUS

The SRI FASTUS system is based on a series of finite-state transducers that compute the transformation of text from sequences of characters to domain templates. This architecture has proven to be very flexible, and has been applied with success to a number of different information extraction tasks in widely varying domains. We have applied FASTUS to extraction of information about terrorist incidents [3], extraction of information about joint ventures [2], indexing of legal documents for hypertext, extracting extensive information from military texts (Warbreaker Message Handler), extraction of information from spoken dialogues [4], and a number of other smaller systems and pilot applications. We have applied FASTUS to Japanese texts [2, 4] as well as English.

Each transducer (or "phase") in the series takes the output of the previous phase and maps it into structures that comprise the input to the next phase, or that contain the domain template information that is the output of the extraction process. It is possible to vary the number of transducers as a parameter of an application, as well as to control precisely how each transducer accepts and produces output. A transducer may handle input by nondeterministically starting at each point in the input stream, or sequentially by determining the final states reachable from the first point of the input stream, and then restarting the transducer at the end of each successive "best" analysis. Typically, all FASTUS phases except the final phase follow the latter regimen, and the templates for all the fragments are merged to form the final analysis. Phases also have the option of passing unanalyzable input to the next phase, or eliminating it from the stream.

The MUC-6 system employs the following sequence of transducers:

1. *Tokenizer.* This phase accepts a stream of characters as input, and transforms it into a sequence of tokens. Most English text is tokenized in the same way, so applications that require heavy runtime optimization can replace this phase by one that is coded directly in the implementation programming language. However, some domains that make unusual demands on tokenization, (i.e. the text contains frequent chemical or mathematical formulas, or names with internal structure, like names for chemical compounds or drugs) may require their own tokenizers, and FASTUS makes an excellent rapid-prototyping tool. In Japanese, where tokenization is problematic, we have replaced the tokenization phase by a standard off-the-shelf segmenter (JUMAN). The result of the tokenization is to ignore completely the whitespace in the input text stream. The FASTUS system preserves whitespace information internally to facilitate the analysis of spatially structured objects like tables and outlines, but this capability, much exercised in the Warbreaker Message Handler, was of no consequence for MUC-6.

2. *Multiword Analyzer.* This phase is generated automatically by the lexicon to recognize token sequences (like "because of") that are combined to form single lexical items.

3. *Preprocessor.* The preprocessor is the point at which the application developer can insert a transducer to handle more complex or productive multiword constructs than could be handled automatically from the lexicon. An example is the transformation of a sequence like "twenty three" into a single number, associated with its numeric value.

4. *Name Recognizer.* This phase recognizes word sequences that can be unambiguously identified as names (like "ABC Corp." and "John Smith"). It also finds unknown words and sequences of capitalized words that don't fit other known name patterns, and flags them so that subsequent phases can determine their type, using broader context.

5. *Parser.* This phase constructs basic syntactic constituents of English, consisting only of those that can be nearly unambiguously constructed from the input using finite-state rules. The output of this phase consists of noun groups (the part of the noun phrase from the determiner through the head noun) and verb groups (the verb together with auxiliaries and adjacent and intervening adverbs). Punctuation, prepositions, relative pronouns, and conjunctions are passed through as 'particles.'

6. *Combiner.* The combiner produces larger constituents from the output of the parser when these can be combined fairly reliably on the basis of local information. Examples are appositives, ("John Smith, 56, president of Foobarco"), coordination of same-type entities, and locative and temporal prepositional phrases.

7. *Domain.* The final phase recognizes the particular combinations of subjects, verbs, and objects that are necessary for correctly filling the templates for a given information extraction task. While the earlier FASTUS phases may have minor domain-dependent parts, they are largely domain independent. Before MUC-6 the domain phase of each FASTUS system was entirely domain dependent, and was rewritten from scratch for each application. In MUC-6 we tested a new idea of a "domain-independent" domain phase that can be easily customized to a new domain. This effort is described below.

The basic FASTUS system includes a merger for merging the templates produced by the domain phase. Merging is essentially a unification operation; the precise specifications for merging are provided by the system developer when the domain template is defined. The developer specifies for each slot what type of data is contained in that slot, and for each data type, FASTUS provides procedures that compare two items of that type and decide whether they are identical or necesarily distinct, whether one is more or less general than the other or the two are incomparable. Depending on the results of this comparison, the merge instructions specify whether the objects can be merged, or if not, the candidates should be combined as distinct items of a set, or if the merge should be rejected as inconsistent. The merger makes the assumption that these comparison and merge decisions are context independent, i.e. it is not necessary to know anything other than the values of the slots to determine whether they merge. For MUC-6, we found it desirable to allow limited cross-slot constraints in the form of equality and inequality constraints.

FASTUS FOR MUC-6

The development of FASTUS since its introduction in 1992 has been focused primarily on making the system easier to use and adapt to new domains. The original system demonstrated in MUC-4 used transition tables that were constructed by hand, and its semantics were embodied solely in lisp code associated with the virtual machine states. For MUC-5, we had developed a system that allowed the system developer to encode automata with a graphical user interface that constructed the transition tables. Subsequent to MUC-5 we developed a specification language (called FAST-SPEC) that allows the developer to write regular productions, that are translated automatically into finite state machines by an optimizing compiler.

This last step greatly facilitated the ability to port FASTUS to new domains quickly. The shortcoming remained, however, that writing FASTSPEC rules was not something that one could reasonably expect an analyst to do in response to an information extraction need. If information extraction systems are going to be used in a wide variety of applications, it will ultimately be necessary for the end users to be able to customize the systems themselves in a relatively short time.

Customizing an extraction system to a domain has always been a long and tedious process. One must determine all the ways in which the target information is expressed in a given corpus, and then think of all the plausible variants of those ways, so that appropriate regular patterns can be written. Because computational linguists have been developing systems for a long time that employ grammars that capture the relevant linguistic generalizations, one might be led to believe that systems that are based on linguistically-motivated English grammars would be much easier to adapt to a new domain.

It has, however, been the experience at past MUC evaluations that systems based on general grammars have not performed as well as those that have been customized in a more application-dependent manner. The reasons for this are more practical than theoretical. General grammars of English, by virtue of being general, are also highly ambiguous. One consequence of this ambiguity is that a relatively long processing time is required for each sentence; this implies, in turn, a relatively long develop-test-debug cycle. Moreover, these systems have proved rather brittle when faced with the multitude of problems that arise when confronted by real-world text. (Lack of robustness may not be inherent in the approach, and much of the current work in corpus-based statistical models is an attempt to overcome this problem).

One might naturally wonder whether one can have the advantages of both worlds: tightly defined, mostly unambiguous patterns that cover precisely the ways the target information is expressed, *and* a way of capturing the linguistic generalizations that would make it unnecessary for an analyst to enumerate all the possible ways of expressing it. We feel that the FASTUS system developed for MUC-6 represents a major step toward achieving this synthesis.

In the current FASTUS system, we attempt to localize the domain-dependence of the rules to the maximum extent possible. To this end, the FASTPEC rules of the domain phase have been divided into domain-dependent and domain-independent portions. The domain-independent part of the domain-phase consists of a number of rules that one might characterize as parameterized macros. The rules cover various syntactic constructs at a relatively coarse granularity, the objective being to construct the appropriate predicate-argument relations for verbs that behave according to that pattern. The domain-dependent rules comprise the clusters of parameters that must be instantiated by the 'macros' to produce the actual rules. These domain-dependent rules specify precisely which verbs carry the domain-relevant information, and specify the domain-dependent restrictions on the arguments, as well as the semantics for the rule.

An example of a typical macro rule is the rule called ActiveBase:

```
EVENT-PHRASE --> EVENT-ADJUNCT* (NG[??subj] ({COMPL | COMPL1}))
                        VG[Active=T,Subcat=Basic,??head]
                        (NG[??obj])
                        {P[??prep1] NG[??pobj1] | P[??prep2] NG[??pobj2] |
                         P[??prep3] NG[??pobj3] | EVENT-ADJUNCT}*;
        head = (head 2);
        rule-type = ActiveBase;
        svo-pattern = ??label;
                        ??semantics;;
```

This rule describes the basic subject-verb-object pattern of a simple active-voice declarative sentence with a transitive verb. The EVENT-ADJUNCT non-terminal parses locative and temporal adjuncts (as well as absorbing otherwise unknown constituents). The next optional constituent is the subject noun phrase, which optionally skips any complements that may be present, followed by an active verb, an optional object, and up to three prepositional arguments, optionally interspersed with temporal and locative adjuncts. The alert reader will notice that the only required element in this pattern is the verb—in analyzing a typical sentence, each pattern will be instantiated multiple times as FASTUS nondeterministically ignores or recognizes the various arguments. The preferred analysis is, of course, the one that is the most complete.

The tokens beginning with "??" in the above example are parameters that are specified by the domain-specific rules when the macro is expanded. Thus, this pattern applies only to noun groups meeting the "??subj" constraints, and to verbs meeting the "??head" restrictions, etc.

Currently, domain-specific rules are centered around verbs. In a typical information extraction task, one is interested in events and relationships holding among entities, and these are usually specified by verbs. Verbs, of course, have corresponding nominalizations, so the macros should automatically instantiate nominalization patterns as well. Unfortunately, the current FASTUS lexicon is not rich enough reliably to make the connection between verbs and their corresponding nominalizations, so the FASTUS system employed for the MUC-6 evaluation did not recognize any nominalized events (like "resignation" or "promotion"). This is an example of a large gap that is easy to close.

The success of this general approach depends heavily on two prerequisites: reliable coreference resolution and a well-developed combiner phase. The coreference module is necessary because it relieves the developer of the domain phase rules of the burden of anticipating all the variations that would result from pronominal and definite reference. Otherwise the developer must see to it that every rule that involves a company as subject also applies to "it," when it refers to a company, as well as to "the company," "the concern,", etc. The FASTUS coreference module resolves pronouns, reflexives, definites, and some bare nominal temporal expressions, with simple algorithms. (There is a separate Alias Recognition module that also contributes to the overall coreference output.) The entity associated with an anaphor gets merged with the first consistent entity found while traversing an ordered list of candidate phrases, each of which is associated with a set of entities. Different types of anaphors call for slightly different candidate phrase ordering and consistency checking algorithms. Our focus was on coreference of phrases that referred to individuals, not types, for it is individual coreference that is needed in most information extraction tasks. Type coreference is both theoretically and practically more difficult, as evidenced by the difficulty of reliable bare-nominal resolution, and its utility in information extraction tasks is unclear. Areas of future extensions are intrasentential coreference based on sentence patterns and limited plausibility inferences based on described events.

The combiner has the responsibility of correctly analyzing appositives and noun-phrase conjunction. This makes it possible for the domain phase to skip complements correctly. If all this work is done, then the specification of domain-specific rules can be a surprisingly simple task.

This system of *compile-time transformations* allowed us to cover with 12 macro rules and 15 domain-dependent rules what would otherwise require approximately one hundred patterns, were the patterns to be written out explicitly. (Not every macro rule applies to every domain-dependent rule.) The domain phase for MUC-6 was developed in less than one person-day.

The set of FASTSPEC grammar rules resulting from the application of the domain-independent macros to the domain-dependent parameters are very close to those that a developer would have written, had he or she been encoding them directly. Thus, the macro rules facility preserves the ability to write patterns that are tightly constrained to fit the particular relevant sentences of the domain, but with the additional advantage of automatically generating all of the possible linguistic variations in an error-free manner. A developer need no longer lament having failed to include a 'passive' variant of a particular pattern simply because no instance occurred in the training corpus. Also, the information specified by the domain-dependent rules is relatively straightforward to provide, (although currently obscured by a rather opaque syntax) so that with the help of a suitable user interface, it is easy to imagine an analyst supplying the system with the information needed to customize it to a new extraction task. Developing such tools is one of our next priorities.

OVERALL PERFORMANCE ON MUC-6 TESTS

FASTUS achieved an outstanding result of F 94 (Recall 92, Precision 96) on the named entity recognition task. The scores for the Template Entity task were somewhat lower F 75.0 (Recall 74, Precision 76). This is to be expected, because some of the named entities, such as percentages, are very easy to extract reliably, and some of the fields in the template entity task (e.g. descriptors) are extremly difficult to extract reliably. The system consistently made certain errors in name recognition , and because these culprits popped up often, they had a substantial impact on the score.

- Although there were numerous instances in the test corpus in which “White House” was used to refer straightforwardly to the building, the system always classified it as a government organization.
- Company names that are identical to person names are a frequent source of error. The surname is sometimes categorized as an alias for the person and sometimes as an alias for the company, depending on where the surname appears relative to the person name or company name in the text.
- Newspapers are to be classified as companies only when the name is intended to refer to the publishing company rather than the periodical. We currently have no overall strategy for distinguishing these cases, although we do pick them up as companies if they are involved in succession events in the scenario template task.
- Location names were to be treated as government entities when the intended referent of the name was the government. We made no attempt to do this correctly.
- When two named entities were combined in a phrase like an appositive that is recognized by the combiner, one of the entites would frequently be lost. For example, “John Smith, a Johnson & Johnson vice president,” would lose Johnson & Johnson. This was due to some remaining bugs in the combiner grammar.

FASTUS achieved one of the better results in the coreference task, with Recall of 59 and Precision of 72.

In the scenario template task, SRI’s FASTUS system achieved a score of F 51.0 (Recall 44, Precision 61). The details of the scenario template task are discussed in the following section.

SRI has been involved in information extraction research for over ten years. As mentioned earlier, the FASTUS System has been under development for a little over three years. SRI undertook a substantial effort prior to the MUC-6 evaluation to clean up all of the domain-independent processing phases, so the domain-independent macro rules could be tested and validated. This effort lasted well into the development period for the MUC-6 evaluation. In fact, we were not able to do a scoreable run of the development training corpus until September 22—two weeks before the test. During this period we were able to quickly bring the system from an F-measure of 32.2 to 55.3 the day before the test. Nearly all the development effort was focused on the combiner phase and on merging and coreference. As noted above, the total amount of time spent on domain patterns was less than a day. Examining the results of the test leads us to believe that many of the problems the system encountered represent not conceptual difficulties but easily fillable gaps, such as the nominalization problem referred to above, or missing domain-relevant lexical features on important words, that would disappear with a short period of additional development.

This experience also supports the view that customization of FASTUS to a new domain is relatively easy and thus gives us reason for a good deal of optimisim about the future for practical applications of information extraction technology.

DISCUSSION OF THE EXAMPLE

The difficulty of building an extraction system is determined to a significant extent by the design of the templates to be filled. Ideally, the structure of the templates will correspond in a systematic way to the linguistic structures through which the relevant information is typically expressed in natural language. Unfortunately this ideal is rarely met.

The MUC-6 template for the scenario template task presented certain problems. In particular there was a lack of fit between the conceptualization of succession events embodied in the template and the typical expression of the corresponding events in language. For example, it is often the case that a single event report (e.g. "John Smith left Microsoft to head a new subsidiary at Apple") corresponds to multiple succession events. Conversely, it is (even more) typical to have a single succession event expressed by multiple sentences (events-reports), often far removed from one another. Also, static information (e.g. "John Smith has been chairman for the last five years.") is often essential to filling the final template, although the succession event structure provides no way of representing this static information.

The Representation of States and Transitions

We feel that the proper template design, or *ontology*, is essential for the rapid development of an information extraction application. For this reason we developed our own internal representation of the domain that corresponded more closely with the ways the information is typically expressed in the texts. A post processor was written to generate the official MUC-6 templates from this internal representation.

We felt that a more appropriate representation of the domain involved two kinds of structures: states and transitions. A state consists of the association among a person, an organization, and a position at a given point in time. A transition is a ternary relation between states and reasons, associating a start state and and end state with a transition reason. In what follows, we will use "position" to refer to position-organization pairs.

The system recognizes two kinds of transitions associated with a succession event: a person pivot, which is a transition in which a start state involving a person and a position is related a state involving the same person but a different position, and a position pivot (which is similar to a succession event), which is a transition in which the start and end states involve a single position and two different people. If a sentence directly implies one of these transitions, then transitions of the other type ('shadow' transitions) are also implied. For example, given the sentence "John Smith resigned as executive vice president of Microsoft" the system represents the content of the sentence as a transition involving the state "John Smith, executive vice president, Microsoft" to "John Smith, some other position, some other company." The system then also generates the implied position pivot, namely the transition from "John Smith, executive vice president, Microsoft" to "Some person, not John smith, executive vice president, Microsoft."

The shadow transitions provide a locus for merging of other states and transitions that may be mentioned in the text. For example, if the next sentence were "Joe Schmoe will assume the post of vice president next month," it would produce a shadow position pivot that would merge with the shadow position pivot from the previous sentence. States that are not otherwise associated with transitions can be merged with transitions. If the next sentence were "Joe Schmoe is the new executive vice president," this would also merge with the end state of the shadow position pivot generated by the previous sentence.

Merging

We decided to augment the FASTUS merger, described in Section 2 above, to handle equality and inequality constraints among slots. Position pivots and person pivots come with pre-specified constraints among their slots stating which elements of the participating states have to be the same and which must be different. The merger will refuse to merge two templates for which the equality and inequality constraints are not satisfied by the resulting merge. This feature, preventing sparsely instantiated templates from overmerging, has now been incorporated into the general FASTUS merger.

The Walkthrough Example

The official score for FASTUS on the walkthrough message was Recall 50, Precision 60. FASTUS did about as well on this message as on the test as a whole, which implies that this was a fairly typical message, at least as far as the system's processing was concerned.

It is thus quite instructive (even to us) to examine the system's response.

The key postulates three succession events for the text: James out, Dooner in as CEO of McCann-Erickson, James out, Dooner in as chairman of McCann-Erickson, and Kim in as vice chairman of McCann-Erickson.

FASTUS missed the transition event regarding the chairmanship of McCann-Erickson. The key sentence, in paragraph 2, where this was introduced was misanalyzed due to a simple bug in the lexicon. The succession event involving Kim was missed for the simple reason that the verb "hire" was never considered as a domain-relevant verb. There is no conceptual problem here—this is merely a consequence of the short development time available. Adding a subject-verb-object pattern *"Company* hires or recruits *person* from *company* as *position"* and one more small gap in the system's coverage is filled.

What was more disturbing was the second overgenerated succession event found by FASTUS, which was a succession involving Dooner out and Alan Gottesman in as president of Paine Webber. Inspection of the text reveals that Alan Gottesman was mentioned as an analyst with Paine Webber, and was not involved in any succession events. Closer analysis reveals precisely what happened: one sentence in the text is "There are no immediate plans to replace Mr. Dooner as president." The subject of the sentence did not receive a domain analysis, bug the verb phrase "replace Mr. Dooner as president" did receive an analysis and produced a partially instantiated position pivot transition with Dooner as president of something being replaced by somebody else as president of something. The mention of Alan Gottesman as an analyst at Paine Webber produced a state (not associated with any transition) consisting simply of Alan Gottesman and Paine Webber (since the position "analyst" was not a high corporate officer, it was simply ignored, and the position in the template left uninstantiated). When merging took place, this state merged with the sparsely instantiated end state of the position pivot, filling out the overgenerated transition and leading eventually to the incorrect succession event.

We were dismayed to discover what appeared to be a grevious but previously undetected bug: sparsely instantiated states and transitions were being allowed to merge, producing many spurious results. This bug was fixed by establishing some minimal instantiation requirements for state to transition merges, and we reran the test and rescored the results. We discovered that our score with the 'bug' fixed was F 47.6, (Recall 36, Precision 69). This bug had purchased us an increase of nearly 4 points in F measure.

While it is tempting at this point to relabel the 'bug' as a 'feature' and consider the matter no further, there is actually a rather interesting story to be told as to why our performance was helped so much by this bug, a story that suggests interesting lines for further investigation.

High Recall, Low Precision Extraction

Hardly anyone has attempted to develop a high-recall low-precision extraction system. Part of the problem is that it is far from clear how to go about doing it. Typically, extraction systems are built by implementing some likely domain-relevant patterns that signal important information in the text, and then examining ever more texts to find the ever less frequent patterns that signal task relevance. This procedure naturally approaches the problem from the low-recall, high precision side. The first patterns that come to mind are likely to be the most reliable. As you add more and more of the rare ones, eventually precision declines as recall creeps upward.

But, what if one wanted to approach the problem from the other angle? The basic idea would be the following: posit every entity of the right type as a candidate for participation in one of the events/relationships of interest, merge to produce more fully instantianted events/relationships and then filter according to some application-specific criteria. It is plausible to suppose that one would start with fairly high recall and gradually, by developing better filter criteria, one would eliminate most of the clearly irrelevant hypotheses, while eliminating few of the relevant ones.

This is a quite reasonable approach for certain extraction tasks, even those tasks for which high recall and low precision is not an acceptable tradeoff. Such tasks are characterized by the following features: (1) entities in the domain have easily determined types and (2) the templates are structured so that there is only one or a very small number of possible slots that an entity of a given type can fill and only entities of a given type can fill those slots. The microelectronics domain of the MUC-5 evaluation [6] was a good example of a domain with these characteristics, and techniques similar to these were successfully applied by at least one system in that evaluation [5]. Our own experience in working in the labor negotiation domain of the MUC-6 dry run has suggested that that domain was also reasonable to approach from this standpoint.

We attempted to develop a system that approached the succession task in this manner. We called this approach the ‘atomistic’ approach or the ‘one rule’ approach, because it was based on finding distinct atoms of relevant information and it was implemented by a single domain rule in FASTUS. This single rule would look for any PERSON, COMPANY or POSITION in the text, and hypothesize a transition event involving that entity. These typically very partial transitions would be merged and finally a post processor would be invoked to filter the resulting hypothesized transitions according to various experimental criteria.

After experimenting with this approach for a while, it seemed to us that it would be difficult to raise the F-score beyond the low 40s. The regular (‘molecular’) FASTUS approach with the macro-expanded domain rules was already doing as well in tests and it appeared to have more promise. We began devoting all our efforts to it.

We did realize that the two approaches raised interesting questions, however. In particular, if one has results from both high-recall and high precision systems, can these be combined in some way to produce a result that would be better than either system taken on its own? The answer was by no means obvious, and in the end we put aside both the atomic approach and any attempt to combine the results.

One way to view the bug we discovered in our system is that it accomplishes just that: the bug embodied, quite accidentally, a not unreasonable strategy for selectively adding information to the result, even though the domain phase did not detect a transition involving the entity. Although there was not enough information to actually determine what states the transition applied to, FASTUS was extracting just enough information from the text to conclude that there was a transition. The system then picked some state to instantiate the transition, and this state was both (1) mentioned in general textual proximity to the transition, and (2) not involved in any other known transition event. Although this occasionally produces ridiculous hypotheses, it is frequently correct; transition events are often mentioned in texts in clusters, and the proximity heuristic works well.

ASSESSMENT OF THE RESULTS

We were generally pleased with the results of FASTUS in this evaluation. Our name recognition was close to the best of the among the participating systems and is approaching the practical maximum performance level for this task. Our coreference module performed the best among all the participants. More important, the module played an important role in the scenario template system, and plays an important role in enabling the system to be easily customized to new domains.

The results in the scenario template evaluation were acceptable and analysis of the particular problems encountered reveals that there are still large gains in performance to be had by simple, straightforward hill climbing on training texts.

One of the most promising results of our MUC-6 preparation effort is that we have implemented a complete extraction system using the macro rules that we proposed a year ago. It allows a significant localization of the domain dependence of the system and this is an essential step toward enabling customization of the system by its end users.

As we mentioned in this article, the amount of time spent analyzing and implementing domain patterns for this evaluation was very minimal—a little more than half a day. Given that most of the effort required to develop the domain independent parts of the system to support the macro rule approach has already been done, if we were to repeat a similar domain task, we suspect that much higher performance could be achieved with much less effort.

How successful were we in isloating domain dependence? There were still a few parts of the larger FASTUS system that had to be modified in response to this task. The combiner rule for recognizing appositives had to be modified, because of the frequency of patterns like "John Smith, 56, president of Foobarco,..." Phrases representing positions were marked, but this marking can be derived from features on the head noun. We modified the FASTUS merger to include the equality and inequality constraints, but, as suggested above, this requirement is likely to be useful in implementing other domains as well, and will be retained as part of our basic system.

FUTURE DIRECTIONS

Our experience from MUC-6 suggests two promising areas for further work. The first area is that of tool development to facilitate the customization of the system by analysts. We have developed the underlying infrastructure required to make this possibility a reality, and we now have the capability to begin experimenting with strategies for specifying patterns, and learning patterns from examples.

The other area of research suggested by our serendipitous bug is to investigate more principled means for combining the results of low-recall high-precision analysis, and high-recall low-precision analysis. Our experience in this evaluation suggests that there may be strategies based on partial information, and textual proximity that yield promising results, particularly for applications in which some sacrifice of precision for increased recall is reasonable.

ACKNOWLEDGEMENTS

This research was supported by the Advanced Research Projects Agency under contract N66001-94-C-6044 with NCCOSC, and contract 94-F-1577-00-000 with ORD.

References

[1] Appelt, D. et al., *FASTUS: A Finite-State Processor for Information Extraction from Real-World Text,* Proceedings of the 13th International Joint Conference on Artificial Intelligence (IJCAI-93), August, 1993, pp. 1172–1178.

[2] Appelt, D. et al., *Description of the JV-FASTUS System Used for MUC-5* Proceedings of the Fifth Message Understanding Conference (MUC-5), August 1993, pp. 221-235.

[3] Hobbs, J. et al., *Description of the FASTUS System Used for MUC-4* Proceedings of the Fourth Message Understanding Conference (MUC-4), June, 1992, pp. 268–275.

[4] Kameyama, M and I. Arima, *A Minimalist Approach to Information Extraction from Spoken Dialogues,* Proceedings of the International Symposium on Spoken Dialogues (ISSD-93), pp. 137–140.

[5] Dekang Lin, *Description of the NUBA System as Used for MUC-5* Proceedings of the Fifth Message Understanding Conference (MUC-5), August, 1993, pp. 263–275.

[6] Onyshkevych, B. et al., *Tasks, Domains, and Languages* Proceedings of the Fifth Message Understanding Conference (MUC-5), August, 1993, pp. 7–18.