ADVANCED DECISION SYSTEMS: DESCRIPTION OF THE CODEX SYSTEM AS USED FOR MUC-3

Laura Blumer Balcom
Richard M. Tong
Advanced Decision Systems
1500 Plymouth Street
Mountain View, California 94043
lbalcom@ads.com
rtong@ads.com
(415)-960-7300

BACKGROUND

ADS has been developing an approach to text processing, called CODEX, COntext directed Data EXtraction), that couples a concept-based, probabilistic keyword pattern matcher, called RUBRIC, with a probabilistic, generalized graph composition chart parser, called CAUCUS. We configure these two key technologies together for a variety of natural language sorting and gisting applications to provide greater depth of analysis (higher precision) than keyword-based techniques alone, as well as higher throughput and greater breadth of coverage than parsing techniques alone.

In a typical text data extraction application, a complete syntactic, semantic, and pragmatic analysis of relevant text segments is required in order to achieve the precision necessary to reduce the level of human interaction required for reliable system performance. This is the role of CAUCUS. Unfortunately, both the knowledge base development and the computational costs are currently too high to apply this technique indiscriminately to all of the text entering a typical data extraction system, where many documents may contain no relevant information, and potentially large segments of relevant texts are also uninteresting. Coupling the keyword pattern matcher with the parser appropriately minimizes the size of the required parsing knowledge bases as well as the amount of text that needs to be interpreted in detail. In future research we expect to show that RUBRIC can improve CAUCUS' performance even further by altering *a priori* confidences in various choices considered by the parser, based on the most probable concepts instantiated by the keyword processing.

Prior to MUC-3 and apart from the development of RUBRIC for its original IR function, approximately five staff years have gone into developing the CODEX system and CAUCUS knowledge bases used in MUC-3. A prototype implementation of CODEX has been demonstrated in a military message domain, and an operational version of that system has been partially deployed. The operational system, written in C to run efficiently on a Mac II, as a runtime system only, could not be used for MUC-3 because it was not finished and because it was not designed to handle the knowledge engineering tasks. Thus, the MUC-3 system evolved from the prototype implementation of the military message handling system. Originally written in Allegro Common Lisp to run on a Sun3 workstation, the CAUCUS module of this system was designed to accommodate knowledge engineering and NLP research, rather than processing efficiency. Unfortunately, its inefficient use of memory had to be remedied in order to compile a 10,000 word lexicon and parse an average MUC-3 sentence within a reasonable timeframe. Though we did eventually solve this problem, we did not do so in time to get the reengineered CAUCUS to produce output for the official MUC-3 testing. Our MUC-3 results thus reflect only the output of a Profiler configured to find relevant text for the parser to analyze. It is important to note that the output generated for MUC-3 official testing strongly reflects our Profiler/Analyzer strategy; we could have extracted more template slot fillers from the Profiler output than we did, but we did not because we intended to have the Analyzer produce these slot fillers.

SYSTEM ARCHITECTURE

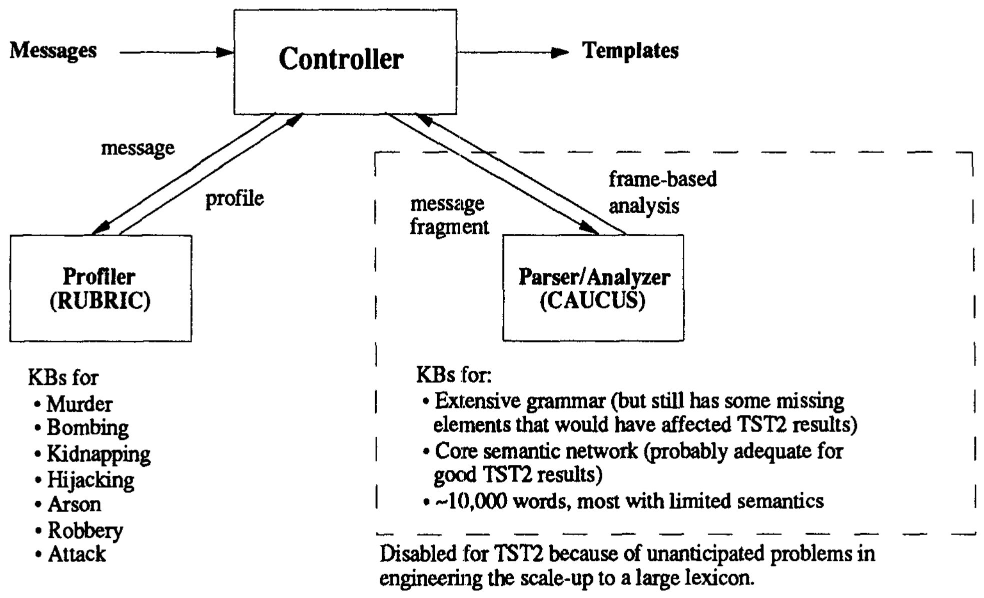

Messages
Controller
Templates
message
profile
message fragment
frame-based analysis
Profiler (RUBRIC)
Parser/Analyzer (CAUCUS)
KBs for
• Murder
• Bombing
• Kidnapping
• Hijacking
• Arson
• Robbery
• Attack
KBs for:
• Extensive grammar (but still has some missing elements that would have affected TST2 results)
• Core semantic network (probably adequate for good TST2 results)
• ~10,000 words, most with limited semantics
Disabled for TST2 because of unanticipated problems in engineering the scale-up to a large lexicon.

Figure 1: CODEX - Status for MUC3 Phase 2

The CODEX architecture is depicted in Figure 1. This architecture separates linguistic and conceptual domain knowledge (in the Profiler and Analyzer) from the particular information requirements of an application (in the Controller) so that these knowledge bases can be used as interchangeable building blocks, reducing the incremental cost of adding new domains, languages, and applications. We estimate that at least eighty percent of our lexicon development time on MUC-3 was spent creating lexical entries that are generally applicable to any NL parsing problem. Only the proper nouns do not have general applicability.

The Controller is an expert system that knows about document formats and the data templates to be filled. It transforms and marks up the input stream into a canonical format for further processing, extracts data from formatted fields that do not require language analysis, sends it to the Profiler for keyword analysis, extracts relevant information from the concept profile, sends relevant fragments to the Analyzer, extracts relevant information from the text fragment interpretation, and puts the extracted information into the required output format.

The Profiler, based on ADS' RUBRIC information retrieval technology, takes the complete text of a document and returns a profile indicating the Profiler's confidence in the presence of each relevant concept, as well as the location of textual keyword evidence for the concept.

The Analyzer, which was disabled for official MUC-3 testing, performs detailed analysis of text to a depth required to completely disambiguate and interpret it within the bounds of a specific domain and application. In its current MUC-3 configuration, input to the Analyzer would have been a sentence to be analyzed had CAUCUS processing been turned on. As we have not yet developed any linguistic modules for inter-sentential analysis, we did expect this omission to have an impact on our MUC-3 scores. CAUCUS is designed so that it can also take as input a set of contextual constraints and hypotheses about the content of the input text as determined by the Profiler and Controller. We plan to use this feature to test the idea that the CAUCUS can be made to produce a more accurate analysis in a shorter time by using this additional input.

Only text fragments that require depth analysis for the needs of the application are passed to the Analyzer, as the deeper analysis is necessarily slower than the more shallow analysis. ADS' Analyzer, called CAUCUS, is only partially implemented to date. Currently it consists of a parser that does both syntactic and semantic analysis. The complete CAUCUS concept also has various asynchronous pragmatic processes such as discourse focus tracking and plausibility analysis interacting with the parser through a global chart. The syntactic and semantic knowledge bases for the parser are declarative and modular, making it possible to interchange domain-specific modules and opening up the possibility of automatic and interactive incremental knowledge acquisition. The modules work together under a best-first chart parsing strategy that optimizes speed without sacrificing recovery from unexpected input.

CAUCUS is based on the PATR family of graph unification chart parsers. It can be configured to behave like a PATR parser, using graph unification and a top-down, left-to-right parsing strategy to find all possible parses of an input. CAUCUS uses PATR-like specifications for rules, lexical entries, and templates, so that a PATR grammar and lexicon could be implemented in CAUCUS in a straight-forward manner. CAUCUS' extensions to PATR thus retain the modular, declarative representation of linguistic knowledge as feature sets, but they provide for a great deal of flexibility for exploring ways of improving the parser's scalability and robustness in the face of apparent ambiguities and unexpected input.

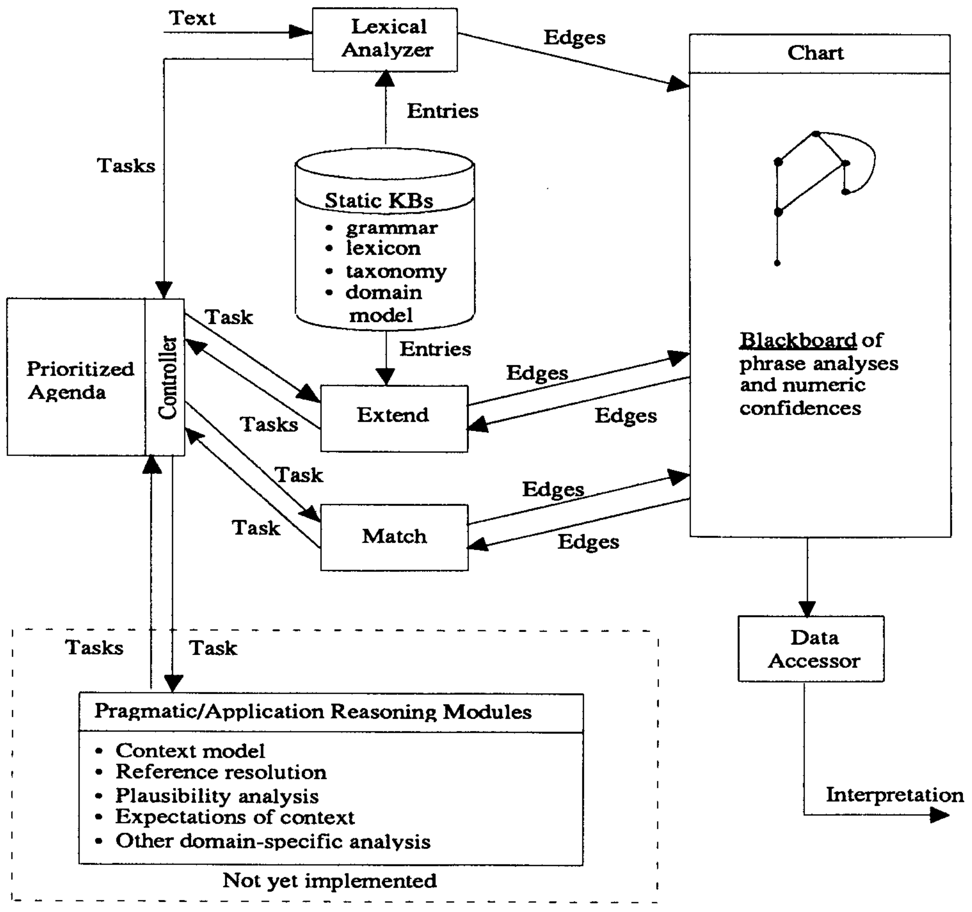


Figure 2: CAUCUS Architecture

CAUCUS' architecture is depicted in Figure 2. As in the PATR parsers, CAUCUS has a chart of edges, which contain directed graph representations of the phrase analyses that they represent. Also like the PATR parsers, edges can be extended and/or matched to create longer edges. Unlike PATR parsers, though, instantiation of extend and match tasks are decoupled from their execution, allowing the strategy for task selection to be determined by a separate control function. This control function manipulates the placement of tasks on the prioritized agenda by giving them numeric priorities reflecting the likelihood that executing the task next will result in reaching the correct parse in the shortest time. Also unlike PATR parsers, chart edges have numeric confidences reflecting the likelihood that the associated edge is part of the correct parse of the input string. These confidences come from (1) the distribution of grammatical constructs in a representative corpus of text, (2) the degree to which the input matches the structure generated by the combination of rules and lexical entries combined by the edge, and (3) the degree to which pragmatic processes confirm the phrase meaning as composed by the edge. As this latter source of numeric confidences might imply, another significant difference between CAUCUS and the PATR parsers is the addition of pragmatics tasks to the parsing agenda. Finally, a unique feature of the CAUCUS architecture is the generalization of graph unification to allow for non-Boolean and non-string-matching composition functions. We call this generalization of Unification Grammar "Generalized Composition Grammar" or "GCG."

To date, we have implemented composition functions in CAUCUS for nodes in a class lattice and nodes representing the semantics of conjunctions, as well as the usual string match of straight graph unification. In addition to these, we plan to implement composition functions that perform reference resolution, spatial and temporal reasoning, part-whole reasoning, reasoning about measures, and other such special purpose semantic and pragmatic functions which do not translate well to traditional feature structure unification. Some of these composition functions will certainly yield non-boolean results on the question of whether two nodes are composeable. In addition to the node composition functions, we also plan to implement a probabilistic feature set composition function, where the degree of match between a top-down proposed structure and its constituents composed from the bottom up is inversely proportional to the probability that constraints violated by a given input structure tend to be violated over a representative corpus of text.

CAUCUS couples a semantic taxonomy with a declarative grammar and lexicon. The grammar/lexicon specification is similar to Lexical Functional Grammar in that compiled entries are used by the parser to build distinct syntactic constituent and functional structures for an input text segment, with the mapping between the constituent and functional structure specified by the lexical entries. In addition, CAUCUS simultaneously composes a semantic functional structure based on the semantic functional structure and selectional restrictions specified in the lexical entries and semantic taxonomy. This knowledge base structure was designed for maximum portability to new discourse domains, languages, and applications. For portability to new discourse domains, we maintain a core grammar, lexicon, and taxonomy, to which we add domain-specific language constructs and concepts. Most of the lexical entries developed for MUC-3 are generally applicable to any parsing problem and have been added to the core.

CAUCUS' probabilistic parsing strategy is similar to deterministic parsers in that the parser is directed to find the most probable solution first, but it differs in two critical ways. First, probabilistic constraint relaxation is built into a function that determines the degree of match between the input and a hypothesized interpretation structure. The degree of match is then rolled into the probability of continuing along the same search path. This enables the parser to exhibit both speed and robustness in the face of unexpected input. Second, instead of a procedurally determined, hardwired parse strategy, ADS' parsing strategy is determined primarily by usage frequencies stored with the rules and word senses retrieved by the parser as they are needed. Thus, the parse strategy can be tuned dynamically to different situations. Currently, the ability to use usage frequency has been implemented, but non-Boolean matching has not.

Probabilistic best-first chart parsing, made possible by our Generalized Composition Grammar and the CAUCUS architecture, is designed to improve the scalability and robustness of natural language parsing. CAUCUS is an active chart parser because phrase constituents are generated once only and made available to be matched with adjacent (active against inactive) edges that might be generated in the future. Typically, a chart parser is used to generate all possible parses in an efficient manner, so the active chart edge matching and proposing mechanisms are tightly coupled in a deterministic top-down, left-to-right control mechanism. This is a sensible approach for testing the generative and string recognition power of a grammar, but for a parser doing real text understanding tasks, the goal is to find the best fit that the non-deterministic knowledge bases can make to the input string in the shortest time possible. In CAUCUS, tasks are placed on a prioritized agenda as they are instantiated by the execution of other tasks. Executing a task may cause the instantiation of any number of tasks, from zero to many, and where they are placed on the queue is a function of an estimated likelihood that executing the task next will cause the parser to discover the best interpretation of the input in the shortest amount of time.

In CAUCUS, there are currently two types of tasks, EXTEND an active or inactive edge, and MATCH an active edge with an inactive edge, either to the right or to the left. Various parameters may be set to manipulate which tasks actually get instantiated during the execution of one of these tasks, and the composition functions and the functions that calculate the priority of a task can all be substituted freely without adversely affecting the parser's operation. Currently, matching is decomposed into two stages in order to minimize the number of graph compositions that are actually executed. As the number of specialized composition functions and pragmatic functions is increased, it may become appropriate to break some of these out as additional task types.

In summary, ADS is developing a unique approach to natural language analysis, called CODEX, that combines a probabilistic, concept-oriented keyword pattern matcher with a probabilistic, best-first, generalized graph composition, active chart parser. These techniques were designed to address problems limiting the robustness, scalability, and portability of current techniques for data extraction from text. Although we have not completed the research and development on these techniques, we have been implementing them in an incremental fashion as modifications to existing prototypes so that we can document the performance gains of each addition over the older techniques from which these new ones have grown. Because the large MUC-3 lexicon required an upgrade to the parser that we were unable to implement in time, the system we used for MUC-3 testing had the parser disabled.

FLOW OF CONTROL

In this section we show how CODEX minus CAUCUS produced templates for the MUC-3 test message TST1-MUC3-0099.

When the controller is invoked on a series of messages, it first breaks up the messages into a series of files, one message per file. Then it breaks up each of these files into a dateline file and a message body file. Each message body file is handed to the profiler, which uses its knowledge base to locate relevant concepts in the text. The Profiler knowledge base consists of two parts. The first part is a set of RUBRIC rules, and the second part is a set of concepts that the profiler is to report on. As can be seen in Figure 3 the rules may be invoked in a hierarchical manner. Although some of the lower level rules corresponded to slot fillers, as may be seen in Figure 4, we did not use this information to create template fills because our strategy was to have CAUCUS do this by analyzing sentences found by the profiler. For the MUC-3 testing, we configured the profiler to report on the sentences that contained concepts of the various incident types. The strategy was to overgenerate so that CAUCUS could make the final determination of relevance and slot fills. The profiler saves its output to a file for future use by the Controller. Figure 5 shows the message with those words highlighted that triggered profiler rules.

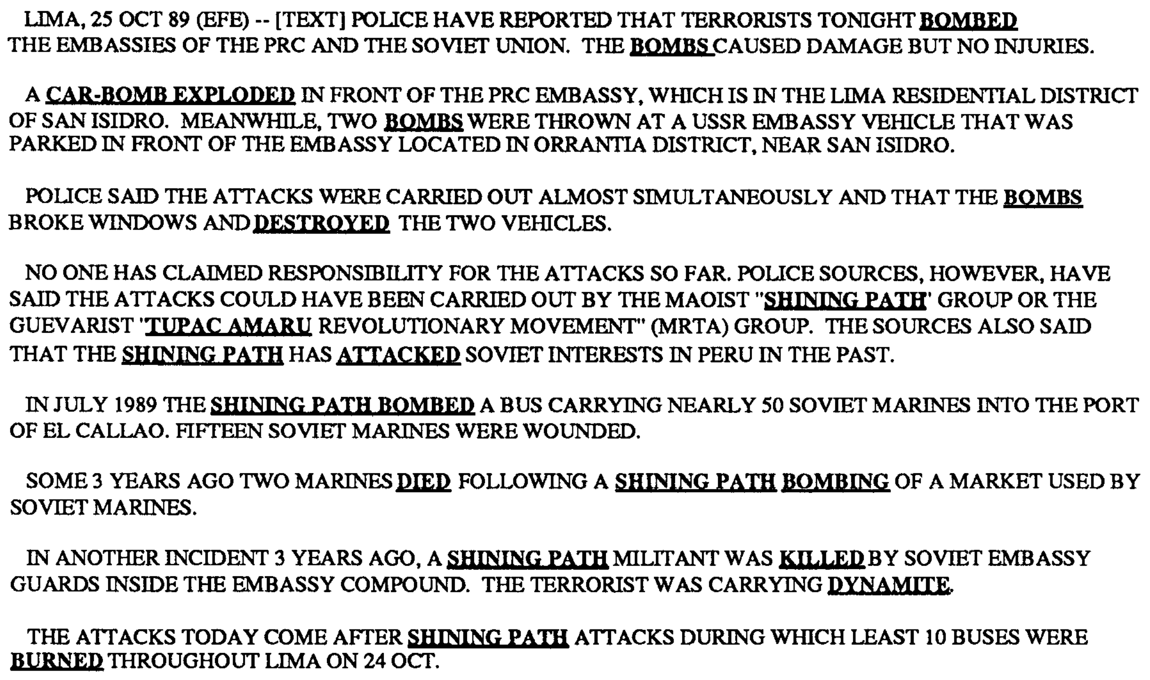
Once the Profiler is finished with a message, the controller loops through the incident types and generates a template for each sentence that contains that incident. For TST1, we generated just one template per incident type concept found in a message, but we found that recall would be higher if we produced one template per sentence containing

```
:IMPLIES
BOMBING-EVENT
:DEFINITION
(:OR (:AND BOMBING DESTRUCTION) (:AND BOMBING CASUALTIES))
:WEIGHT
100
:END-DEFINITION

:EVIDENCE
BOMBING
:DEFINITION
(:OR (:PHRASE "BLOWN" "UP") (:PHRASE "BLEW" "UP")
     "BOMBING" "BOMBINGS" "BOMBED" "DYNAMITED")
:WEIGHT
100
:END-DEFINITION

:EVIDENCE
DESTRUCTION
:DEFINITION
(:OR "DESTROYED" "DAMAGED" (:PHRASE "BLOWN" "TO"))
:WEIGHT
60
:END-DEFINITION
```

Figure 3: Example Profiler Rules

BOMBING	(S1 100) (S3 80) (S9 100) (S11 100)
ATTACK	(S8 100)
EXPLOSIVE-DEVICE	(S2 100) (S3 100) (S4 100) (S5 100) (S13 100)
DESTRUCTION	(S5 60)
CASUALTIES	(S12 50)
DEATH	(S11 100)
KILLING	(S12 100)
BURNING-RESULT	(S14 100)
SHINING-PATH	(S7 100) (S8 100) (S9 100) (S11 100) (S12 100) (S14 100)
TUPAC-AMARU	(S7 100)
BOMBING-EVENT	(S1 60) (S3 48) (S5 36) (S9 60) (S11 60)
ATTACK-EVENT	(S8 70)
MURDER-EVENT	(S12 60)
TERRORIST-EVENT	(S1 48) (S3 38) (S5 29) (S8 70) (S9 60) (S11 60) (S12 60)

Figure 4: Profiled TSTI-MUC3-0099


LIMA, 25 OCT 89 (EFE) -- [TEXT] POLICE HAVE REPORTED THAT TERRORISTS TONIGHT **BOMBED** THE EMBASSIES OF THE PRC AND THE SOVIET UNION. THE **BOMBS** CAUSED DAMAGE BUT NO INJURIES.

A **CAR-BOMB EXPLODED** IN FRONT OF THE PRC EMBASSY, WHICH IS IN THE LIMA RESIDENTIAL DISTRICT OF SAN ISIDRO. MEANWHILE, TWO **BOMBS** WERE THROWN AT A USSR EMBASSY VEHICLE THAT WAS PARKED IN FRONT OF THE EMBASSY LOCATED IN ORRANTIA DISTRICT, NEAR SAN ISIDRO.

POLICE SAID THE ATTACKS WERE CARRIED OUT ALMOST SIMULTANEOUSLY AND THAT THE **BOMBS** BROKE WINDOWS AND **DESTROYED** THE TWO VEHICLES.

NO ONE HAS CLAIMED RESPONSIBILITY FOR THE ATTACKS SO FAR. POLICE SOURCES, HOWEVER, HAVE SAID THE ATTACKS COULD HAVE BEEN CARRIED OUT BY THE MAOIST "**SHINING PATH**" GROUP OR THE GUEVARIST "**TUPAC AMARU** REVOLUTIONARY MOVEMENT" (MRTA) GROUP. THE SOURCES ALSO SAID THAT THE **SHINING PATH** HAS **ATTACKED** SOVIET INTERESTS IN PERU IN THE PAST.

IN JULY 1989 THE **SHINING PATH BOMBED** A BUS CARRYING NEARLY 50 SOVIET MARINES INTO THE PORT OF EL CALLAO. FIFTEEN SOVIET MARINES WERE WOUNDED.

SOME 3 YEARS AGO TWO MARINES **DIED** FOLLOWING A **SHINING PATH BOMBING** OF A MARKET USED BY SOVIET MARINES.

IN ANOTHER INCIDENT 3 YEARS AGO, A **SHINING PATH** MILITANT WAS **KILLED** BY SOVIET EMBASSY GUARDS INSIDE THE EMBASSY COMPOUND. THE TERRORIST WAS CARRYING **DYNAMITE**.

THE ATTACKS TODAY COME AFTER **SHINING PATH** ATTACKS DURING WHICH LEAST 10 BUSES WERE **BURNED** THROUGHOUT LIMA ON 24 OCT.

Figure 5: Scanned TSTI-MUC#-0099

an incident. We also could have configured the profiler to scope concepts at the paragraph level, but with the newswire articles, this would produce very little difference in the result. If any of the incident types are found in a message, the Controller also sends the name of the message file and the potentially relevant sentences to CAUCUS through another file, as shown in Figure 6.

Generating templates based on Profiler output only was a fail-safe mechanism in case of parser failure. As it turned out, this was our only output. Thus, final output for TST1-MUC3-0099 consists of five templates with only the message-id, template-id, and incident-type slots filled out. A BOMBING template is produced for each of the first three sentences in Figure 6, a MURDER template is produced for the fourth sentence in Figure 6, and an ATTACK template is produced for the last. Given these sentences as input, the parser's analysis would have indicated that only the first sentence contained relevant incidents, but it would have created two BOMBING templates, one for each of the physical targets. Before processing the first sentence for the message is a line indicating the number and the dateline file of a new message. The parser uses this information to reset some global variables containing frames representing the dateline information, which it may need to determine the time and location of incidents. With the parser up and running as expected, then, the final output for this message would have been two bombing templates with the date, perpetrator-id, physical-target, foreign-nation, and location slots filled in.

Since we never generated CAUCUS output for this message, we will not go into CAUCUS processing. Our simple initial strategy was to have CAUCUS parse the sentences found by RUBRIC to have concepts of MUC-3 incidents, determine relevance, and change the template output, as described above. This strategy would have missed the effects mentioned in the second sentence of the message, the neighborhood-level locations and secondary physical target mentioned in the second paragraph, or the suspected perpetrator organization mentioned in the third paragraph. In the future we will be experimenting with a feedback loop between the Controller and CAUCUS to provide additional sentences to CAUCUS for analysis, depending on the absence of information in the CAUCUS template output. Thus, only the minimum number of sentences would be parsed. In this case, all but two small sentences would be parsed, but

```
/prj/nlp/muc/picasso/F0000

POLICE HAVE REPORTED THAT
TERRORISTS TONIGHT BOMBED THE EMBASSIES OF THE PRC AND THE SOVIET
UNION.

IN JULY 1989 THE SHINING PATH BOMBED A BUS CARRYING NEARLY 50 SOVIET
MARINES INTO THE PORT OF EL CALLAO.

SOME 3 YEARS AGO TWO MARINES DIED FOLLOWING A SHINING PATH BOMBING OF
A MARKET USED BY SOVIET MARINES.

IN ANOTHER INCIDENT 3 YEARS AGO, A SHINING PATH MILITANT WAS KILLED
BY SOVIET EMBASSY GUARDS INSIDE THE EMBASSY COMPOUND.

THE SOURCES ALSO SAID THAT THE
SHINING PATH HAS ATTACKED SOVIET INTERESTS IN PERU IN THE PAST.

:END-MSG
:END-PROC
```

Figure 6: TST1-MUC3-0099 Input to CAUCUS

in other messages, this strategy should reduce the burden on the parser significantly.

SUMMARY

In this paper we have described ADS' CODEX flexible message understanding architecture, which consists of a Profiler, a Parser/Analyzer, and an application-specific Controller. We also described the flow of control in CODEX as it was instantiated for MUC-3, concentrating on the parts that contributed to our official MUC-3 test run. The CAUCUS parser/analyzer is still quite new, and we ran into an unexpected problem in scaling up the lexicon to handle as many entries as were required for MUC-3. Though we eventually solved this problem, we did not do so in time to use it for MUC-3 testing.