ADVANCED DECISION SYSTEMS' CODEX: MUC-3 TEST RESULTS AND ANALYSIS

Laura Blumer Balcom
Richard M. Tong
Advanced Decision Systems
1500 Plymouth Street
Mountain View, California 94043

INTRODUCTION

ADS has developed a general purpose message data extraction system concept, called CODEX (for COntext directed Data EXtraction), that we instantiated for MUC-3 as shown in Figure 1.

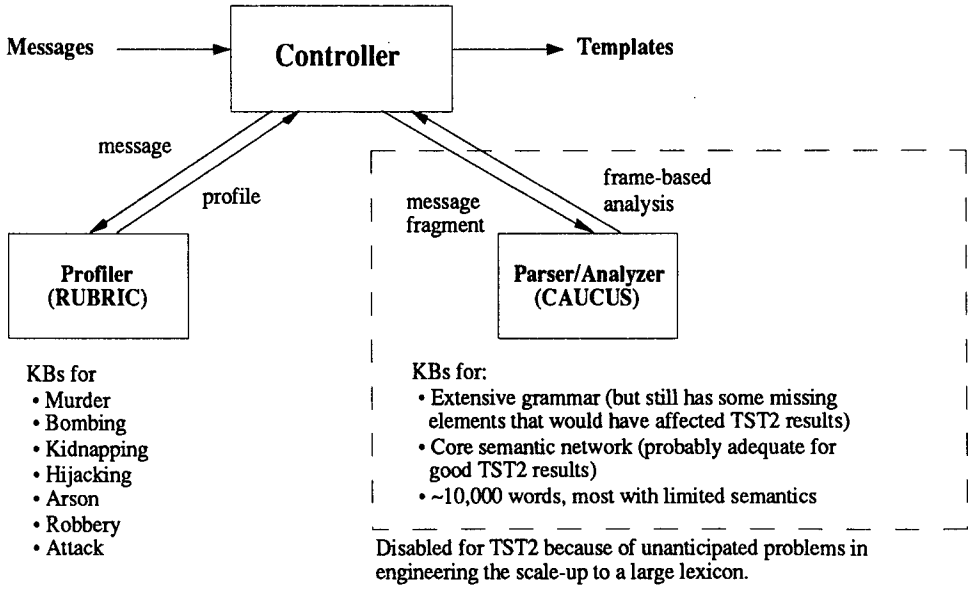

Figure 1: CODEX - Status for MUC3 Phase 2

The underlying principle of the CODEX concept is to do the analysis in two phases. First, we perform a "surface level" analysis of the message text to determine if there is any evidence for data items of interest. If there is not, then no further processing of the message is attempted. If, however, there is evidence for the items of interest, we proceed to the second phase in which detailed analysis is performed only on those sections of the message identified by the first phase as having potentially relevant material in them.

For MUC-3, the profiler was set-up to overgenerate partially completed templates based on the detection of the basic event types (i.e., murder, bombing, kidnapping, etc.) at the sentence level, with the parser/analyzer used to either reject these hypothesized templates or to complete them based on more detailed analysis of the text. The parser/analyzer would also do the appropriate reasoning to determine whether the incident was attempted, threatened, or accomplished. The generation of templates based solely on profiler output was designed to be a fail-safe feature in the event of parser failure. Since CAUCUS was not enabled for TST2, this fail-safe mechanism provided our only output. It produced one template per incident type per sentence in which a concept of the incident type was found.

MUC-3 RESULTS AND ANALYSIS

Table 1 shows the official Total Slot Scores for TST2-MUC3.

SLOT	POS	ACT	COR	PAR	INC	ICR	IPA	SPU	MIS	NON	REC	PRE	OVG	FAL
template-id	118	221	97	0	0	0	0	124	21	17	82	44	56	
incident-date	114	0	0	0	0	0	0	0	114	4	0	*	*	
incident-type	118	97	75	11	11	0	11	0	21	0	68	83	0	0
category	83	0	0	0	0	0	0	0	83	35	0	*	*	0
indiv-perps	97	0	0	0	0	0	0	0	97	54	0	*	*	
org-perps	70	0	0	0	0	0	0	0	70	56	0	*	*	
perp-confidence	70	0	0	0	0	0	0	0	70	56	0	*	*	0
phys-target-ids	55	0	0	0	0	0	0	0	55	79	0	*	*	
phys-target-num	38	0	0	0	0	0	0	0	38	80	0	*	*	
phys-target-types	55	0	0	0	0	0	0	0	55	79	0	*	*	0
human-target-ids	152	0	0	0	0	0	0	0	152	22	0	*	*	
haman-target-num	96	0	0	0	0	0	0	0	96	22	0	*	*	
human-target-types	152	0	0	0	0	0	0	0	152	22	0	*	*	0
target-nationality	18	0	0	0	0	0	0	0	18	105	0	*	*	0
instrument-types	25	0	0	0	0	0	0	0	25	93	0	*	*	0
incident-location	118	0	0	0	0	0	0	0	118	0	0	*	*	
phys-effects	37	0	0	0	0	0	0	0	37	93	0	*	*	0
human-effects	56	0	0	0	0	0	0	0	56	83	0	*	*	0
MATCHED ONLY	1237	318	172	11	11	0	11	124	1043	761	14	56	39	
MATCHED/MISSING	1472	318	172	11	11	0	11	124	1278	900	12	56	39	
ALL TEMPLATES	1472	442	172	11	11	0	11	248	1278	2884	12	40	56	
SET FILLS ONLY	614	97	75	11	11	0	11	0	517	566	13	83	0	0

Table 1: Total Slot Scores

Because of unanticipated system engineering issues associated with the scale-up to a lexicon of the size required for MUC-3, we were unable to run the parser/analyzer on the TST2 message set. The results therefore reflect just the output from the profiler. Since this was set up to provide input to the parser (rather than to perform the best template fill possible), our templates have only the *incident-type* slot filled. We could have used profiler output to fill out some of the other slots to some degree of confidence less than could be obtained through a finer-grained analysis, but we did not do this because our strategy was to have the parser fill out these slots. This strategy might change in the future, depending on the robustness of the parser in analyzing MUC-3 texts.

The high overgeneration of templates based on profiler output was expected, given the processing strategy explained above. Most of this overgeneration will be eliminated with the addition of the parser, assuming it can analyze the sentences given to it as input by the controller.

Our analysis of failures to correctly identify templates is summarized in Table 2. The numbers in this table do not always sum to the total for a row because a single failure may have multiple causes.

Failures appear to reduce to three types. First, we could have gained a few more points in both precision and recall of the incident-type slot by correcting some template mappings. Although we did not realize this when we were scoring our results, the overgeneration resulted in some faulty mappings of optional templates to incorrect overgenerated templates.

Scored	Type of Failure				Total
	Profiler KB	Parser Should Fix	Correct w/ Mapping	Partial Correct w/ Mapping	
Missing	18	2	1		21
Incorrect	1	10	3	5	11
Partial credit	4	7			11

Table 2: Failure Analysis on Incorrectly Identified Templates

Second, the addition of the parser should, in theory, fix most of the incorrect and partially correct responses. The parser should, for example, fix some of the partially correct responses by resolving attacks into bombings or arson by recognizing the incident-instrument relation, resolving attacks into murders by recognizing that death was caused by the event, and resolving murders into death threats by recognizing the speech act-event relation. Most of the over-generated templates that resulted in incorrect mappings would also have been eliminated by recognizing that the description is too vague or the event does not meet one of the relevance criteria. We note, however, that the parser will not eliminate all of the overgeneration until we have developed appropriate modules for reference resolution and discourse structure analysis.

A third type of failure, inadequacies of the profiler knowledge base, contributed primarily to missing templates. The types of profiler knowledge base faults are summarized in Table 3.

Fault	Example	Failures	Solution
Misspelled keyword	"kidnapped"	1	parse sentences with unrecognized words
Generic description	"killings of Jesuits"	7	lower detection threshold
Effect implies event	"bodies of the victims"	10	lower detection threshold
Speech act implies threat	"announced...they will destroy"	7	add threat to profile
Instrument implies event	"threatening...with hand grenades"	4	lower detection threshold
Unusual term	"slaying"	1	create concept tree from CAUCUS KB

Table 3: Profiler Knowledge Base Faults

Most of the suggested solutions to these failures will improve recall while reducing the precision of this portion of the system. By increasing the profiler recall at the expense of precision, we add to the number of sentences that must be analyzed by the parser. RUBRIC is designed so that we can easily experiment with this trade-off, but we did not do this because we expected that improvements to the parser at this stage in its development would achieve higher

payoffs in our MUC-3 score than experiments of this sort.

Two failure types, the misspelled keyword and the unusual term, point out what might be considered a flaw in the filter-before-parsing approach. One could, of course, apply the reverse of spelling correction to all words not found in the parser's lexicon to see if we can make them match keywords, or one could apply a spelling-mess-up program to the profiler keywords to catch potentially misspelled keywords in the text, but these are likely to be high-cost, low-payoff solutions. Another solution might be to parse all sentences with unrecognized words. This approach might have a higher payoff because it is likely to uncover sentences that have lists of perpetrators or victims without mention of any event keywords.

Detecting events described with unusual terms is a problem for all keyword-based concept detection techniques, especially if the knowledge bases are developed automatically through statistical techniques. However, we have an idea for developing concept knowledge bases based on the structure of the parser's semantic network and lexicon that may result in a more complete profiler knowledge base than would be feasible using only statistical techniques.

MUC-3 PREPARATION

Effort expended for MUC-3 involved: (1) Development of profiler rules, (2) Development of grammar, lexicon and semantics for the parser/analyzer, (3) System integration and testing, and (4) General administration. Approximate staff-hours of effort expended on these tasks over the period of the MUC-3 evaluation cycle (i.e., December 1990 through May 1991) is shown in Table 4 separated by domain independent activities and MUC-3 specific activities.

Category	Total Labor Hours	
	Domain Independent	MUC-3 Specific
Profiler	0	180
Parser / Analyzer		
Grammar rules	150	0
Lexical entries	475	20
Semantics	100	50
System Engineering	300	200
Administration	0	175
TOTALS	1025	625

Table 4: MUC-3 Effort by Category

Other than the administrative cost of hosting the MUC-3 interim conference, the largest MUC-3 specific tasks were developing the back-end procedures for extracting template fillers from the parser and profiler results. We have ideas for reducing or eliminating these application-specific tasks, which we hope to implement in the next year. Other domain-specific knowledge engineering tasks involved relatively minor additions to the lexicon and semantic network.

The bulk of effort for MUC-3 went into expanding the capacity of CAUCUS for lexical processing. Under system engineering, we added a lexical analyzer and a facility for storing and accessing compiled lexical entries in disk files, as well as a few tools for partially automating lexical acquisition. As can be seen in Table 5, our core lexicon grew from a few hundred entries to about 10,000. Our grammar grew by about 20%, adding relative clauses and conjunction of clauses and adjectival phrases. We also added a few semantic net nodes to handle some new types of violence and some new speech acts.

Knowledge Base	Before MUC-3	Additions	
		Domain Independent	MUC-3 Specific
Profiler rules	0	0	62
Controller rules	0	0	1
Grammar rules	108	20	0
Words	~500	~9000	~1000
Semantic net nodes	~1100	~50	~50

Table 5: Number of Knowledge Base Structures for MUC-3

LESSONS LEARNED

CODEX was designed to scale up to large realistic problems such as MUC-3. The C/Lisp implementation used in MUC-3 was designed to facilitate knowledge engineering and experimentation with parameters and algorithms that will at least partially automate adaptation to new domains and applications. The Profiler component of the system uses the relatively mature RUBRIC technology, but CAUCUS, the Parser/Analyzer component of the system is still in its infancy. Prior to MUC-3, we had implemented CAUCUS' Generalized Composition Grammar, grammar compiler, chart parser with prioritized agenda, and a semantic network and lexicon that covered several small domains that we have worked in the past. During MUC-3, most of our effort was expended in upgrading CAUCUS' facilities for processing lexemes and adding roughly 10,000 lexical entries. Although we were not, in the end, able to get this facility up in time to test the parser on MUC-3 TST2, all of this work will be generally useful to future applications. For ADS then, MUC-3 served as a catalyst in the development of generally applicable language processing facilities for message data extraction applications.

Though the timing was not quite right for us, our analysis of MUC-3 results shows that this type of testing is useful for assessing the capabilities and weaknesses of a message understanding system as a whole and for showing where future efforts will achieve the highest payoff in improving the system's performance. ADS' results validate our CODEX approach to the degree that it was implemented for MUC-3.