UNISYS:
DESCRIPTION OF THE UNISYS SYSTEM USED FOR MUC-3

Carl Weir, Tim Finin, Robin McEntire, and Barry Silk
Unisys Center for Advanced Information Technology
Paoli, Pennsylvania
weir@prc.unisys.com
215-648-2369

INTRODUCTION

This paper describes the Unisys MUC-3 text understanding system, a system based upon a three-tiered approach to text processing in which a powerful knowledge-based form of information retrieval plays a central role. This knowledge-based form of information retrieval makes it possible to define an effective level of text analysis that falls somewhere between what is possible with standard keyword-based information retrieval techniques and deep linguistic analysis.

The Unisys Center for Advanced Information Technology (CAIT) has a long-standing commitment to NLP research and development, with the Pundit NLP system developed at CAIT serving as the Center's primary research vehicle [3]. The Unisys MUC-3 system, however, consists primarily of components that are less than 7 months old and still in a developmental stage. Although the three-tiered processing approach that the MUC-3 system's architecture is based upon includes Pundit as its third level of (linguistic) analysis, the incorporation of Pundit into the MUC-3 system was not achieved in time for the final MUC-3 test in May, 1991. A decision was made to focus on the development of a knowledge-based information retrieval component, and this precluded the integration of Pundit into the prototype.[1] The Unisys MUC-3 system without its linguistic analysis component is depicted in Figure 1. This is the version of the system that was actually used in the MUC-3 test.

APPROACH AND SYSTEM DESCRIPTION

The Unisys MUC-3 system's architecture consists of five main processing components, three of which represent levels of text understanding. An initial preprocessing component transforms texts into an appropriate format for the text understanding components to manipulate. The three text understanding components engage in (1) standard keyword-based information retrieval, (2) knowledge-based information retrieval, and (3) linguistic analysis.[2] A final, template generation component gathers together all the facts extracted from a given text and builds template data structures out of them. These five components are described in more detail below.

A Message Pre-processing Component

The Unisys MUC-3 system's message pre-processing component is a special, low-level processor which parses texts into their component parts and generates output in a form compatible with the KBIRD rule processing system (i.e., as a set of Prolog terms). This processor is a special C program which was generated using an *Application Specific Language* called MFPL (*Message Format Processing Language*) [6]. MFPL was specifically designed as a high-level language for processing the formatted portions of electronic messages. In addition to producing a representation of the text in Prolog terms, this module identifies and encodes sentence boundaries, paragraph boundaries, and the standard formatted portions of the text (e.g., date, time, location, etc.).

[1] If funding is available, we plan to add Pundit to the system in time for the MUC-4 conference (June, 1992).

[2] As pointed out in the introduction, the Pundit NLP system could not be integrated into the MUC-3 system in time for the final MUC-3 test run.

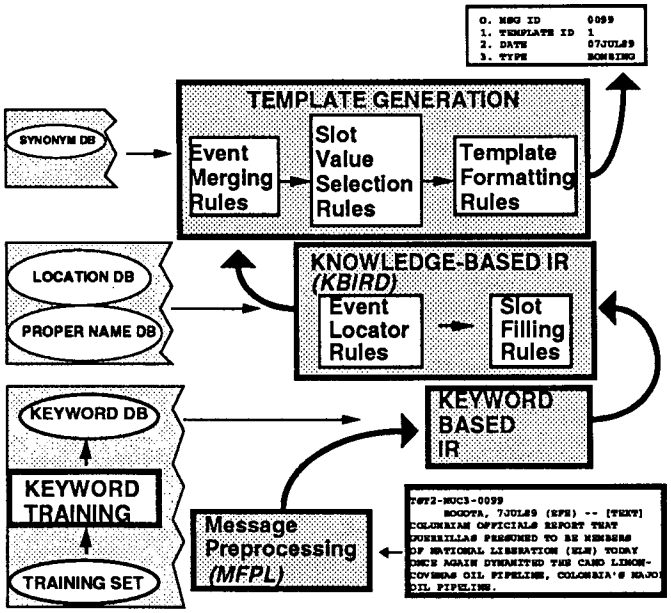

Figure 1: This diagram depicts the version of the Unisys MUC-3 system used in the final MUC-3 test run. The third level of text-understanding in the three-tiered approach to text-processing described in this paper (linguistic analysis provided by the Pundit NLP system) was not incorporated in time for the test, and is therefore not represented in the diagram.

A Keyword-Based Information Retrieval Component

The keyword analysis component of the Unisys MUC-3 system predicts when various types of terrorist acts (bombings, murders, kidnappings, and so forth) have been referred to in a text. The probability of an act of a given type having occurred is determined by a search for words, word stems, and pairs of words and pairs of word stems, that are associated with types of acts.[3] The probability of such a word (or word stem, or word pair or stem pair) occurring in a text for which an act of a given type is associated is determined as follows.

The frequency of presence for a given word W (or word stem ...) in texts for which a terrorist act of a given type T occurs is computed ($f(W,T)$), as is the presence of the word in any text at all in the complete corpus ($f(W,C)$). The probability of the word appearing in texts for which a terrorist act of a given type occurs

$$\frac{f(W,T)}{f(T,C)}$$

and the probability of the word occurring in any text

$$\frac{f(W,C)}{|C|}$$

are calculated, and these two values are used to determine the conditional probability of the word (or word stem ...) predicting the given type of terrorist act.

$$\mathcal{P}(W,T) = \frac{\left(\frac{f(W,T)}{f(T,C)}\right)}{\left(\frac{f(W,C)}{|C|}\right)}$$

[3]The keyword analysis system uses a rule-driven word-stemmer based on one developed by Chris Paice (Landcaster, UK) [4].

Only words with relatively high probabilities of predicting a given type of terrorist act are searched for in a text, and words that do not occur frequently enough in the text corpus based on some empirically-derived threshold are not used.

Training the keyword-based analysis component. A database of key words, two-word phrases, word stems and two-stem phrases was compiled from the DEV corpus using a collection of GAWK scripts. After some experimentation, we decided not to use the word stem and stem-pair data in the final test, because it was not making any positive difference in the system's event detection performance. Currently, an event class, T, is predicted for a text if it contains any single word (or two-word phrase), W, where $\mathcal{P}(W,T) > .65$ or if it contains two words (or two two-word phrases) W_1 and W_2 where $\mathcal{P}(W_1,T) > .55$ and $\mathcal{P}(W_2,T) > .55$. Further experimental variation of the scoring algorithm should result in continued enhancements to this component's event detection capabilities.

A Knowledge-based Information Retrieval Component (KBIRD)

Once a set of terrorist acts have been predicted, the task of generating templates describing those acts falls to the knowledge-based information retrieval component called KBIRD.

KBIRD is a rule-based system for concept-spotting in free text [2, 7]. KBIRD rules are forward-chaining horn clauses whose antecedents are constituents discovered and recorded in a chart data structure and whose consequents are newly inferred constituents—concepts (or facts)—to be added to the chart. The antecedents and consequents of KBIRD rules can include arbitrary Prolog goals just as in *Definite Clause Grammars* [5].

It is tempting to think of a set of KBIRD rules as implementing a kind of bottom up chart parser, but there are several interesting differences. One distinctive feature is that the concepts that KBIRD rules infer are associated with a specific region of text, a region which is the maximal cumulative span of the regions of text associated with each expression in a given rule's antecedent. Moreover, these regions can be explicitly reasoned about by subsequent KBIRD rules.

In typical natural language parsers, there is an implicit constraint that adjacent constituents in a rule must be realized by contiguous strings of text in the input. KBIRD allows one to write rules which specify other constraints on the relative positions of the strings which realize rule constituents. The antecedent of a KBIRD rule may consist of several facts (words or concepts) that are the arguments of operators illustrated below. New operators are easy to define.

Antecedent Format	Operator Description
A ^ B	A is contiguous with B.
A , B	A is in the same text as B.
A .. B	A is in the same sentence as B.
A.. > B	A is in the same sentence as and precedes B.
A ... B	A is in the same paragraph as B.
A ..+ B	A is in the same region as B.

KBIRD rules are compiled into a combination of Prolog backward chaining rules and forward chaining rules in Pfc [1]. A simple optimizer is applied to the output of this compilation process to improve performance. KBIRD has many additional features which are inherited from the Pfc rule language, such as the ability to write non-monotonic rules which specify that no occurrence of a certain constituent or concept be found in a given region.

Some examples of KBIRD rules are shown below. The first rule states that if the wordstem "MURDER*" has been found in the text, then a fact should be added to the factbase stating that a potential murder event has been found. The second rule illustrates KBIRD's ability to recognize phrases, asserting that if the string "ARMY OF NATIONAL LIBERATION" is discovered, a fact should be added to the factbase stating that a terrorist organization exists in the text at the same location as the string. The third rule illustrates the use of operations on concepts derived from the text, asserting that if a terrorist event E is found in the same sentence as a potential victim V, then a fact should be added to the factbase indicating that V is the actual victim of E.

```
1. "MURDER*" ==> potential_murder_event.
2. "ARMY"^"OF"^"NATIONAL"^"LIBERATION" ==> terrorist_organization.
3. terrorist_event(E) .. potential_victim(V) ==> victim(E,V).
```

Several additional features of the KBIRD rule language should be mentioned, all of which appear in the following, more complex rule used to infer individual perpetrators:

```
generic_perpetrator(A)@P,
[~unlikely_perpetrator(Name)],
{get_full_text_at_loc(P,Name)}
     ==> potential_ind_perpetrator(A, Name).
```

In the first clause of the antecedent of this rule, the text location index associated with the concept `generic_perpetrator(A)` is bound to the logic variable `P` with the @ operator. This allows the location to be explicitly constrainted later in the rule. If a clause is enclosed in square brackets, as is the case for the second clause of the antecedent, then its location is ignored. This condition also shows the use of the tilde (~) as a negation operator. Thus, this second clause specifies that it is not the case that `Name` has been determined to be an "unlikely perpetrator" anywhere else in the text. The final clause of the antecedent in this rule is enclosed in curly brackets, which indicates that it is a Prolog constraint which must be met—this clause is used to extract the actual text associated with the concept bound to the logic variable `P`.

A Template Generator

The *Template Generator* has three tasks: to select the actual templates to be produced as output, to choose between candidate slot fillers if more than one has been found, and to print the template in the proper format.

Template Selection. The process of determining which template structures to build out of the facts inferred by KBIRD begins by determining if any events at all have been predicted. If no event has been predicted, then an "irrelevant template" is created. If several events of the same type have been created, the template generator will attempt to merge them using a set of heuristics which hypothesize that two event descriptions refer to the same event. Some of the general heuristics used for merging events of the same class are:

- Merge two events if there is a significant overlap in the text regions found by the *event locator rules.*
- Merge two events if they share human targets whose scores are above a certain threshold.
- Merge two events if they share physical targets whose scores are above a certain threshold.

Slot Filler Selection. After merging events, the template generator must select the final slot filler values. The KBIRD rules which propose slot fillers attach a score (an integer between 0 and 100) to each candidate which represents the system's confidence in that value. If multiple candidate fillers exist for a given template, several general heuristics are used to select among them:

- Candidate slot values with scores below a given threshold are dropped from consideration.
- A set of synonymous expressions are dropped in favor of their canonical expression.
- If one candidate expression is a substring of another, then the shorter one is dropped.
- A generic description (e.g., *vehicles*) is dropped in favor of one or more subsumed ones (e.g., *ambulance, truck*).
- If a slot can only take a single value then the candidate receiving the highest value is selected.

A Linguistic Analysis Component (Pundit)

The Pundit natural language processing system has been under development at Unisys for the last five years and is capable of performing a detailed linguistic analysis of an input text. Unlike KBIRD, Pundit abstracts away from the actual strings used to convey information in a text at the very beginning of its analysis process by determining to which syntactic properties and domain concepts the lexical items in the text correspond. These syntactic properties and domain concepts are then processed without much attention being paid to their physical location in the text. In KBIRD, on the other hand, everything that is manipulated, even concepts that have been asserted, are explicitly associated with regions of text.

A key capability that the deeper linguistic processing of Pundit can provide is the determination of the grammatical and thematic roles of expressions in a text. Thus, it can determine that in the sentence *"Castellar is the second mayor that has been murdered in Colombia in the last 3 days"* that *Castellar* is the subject of the copular verb in the matrix clause, and that *Castellar* should inherit properties asserted of the predicate nominal argument. It can also recognize the passive voice of the relative/subordinate clause headed by *that* and thus that it is Castellar that has been murdered (as the second mayor) in Columbia.

It would be possible to build a KBIRD rulebase that performs the sort of detailed linguistic analysis now being performed by Pundit. Merging KBIRD and Pundit in this way would minimize the complications of integrating the text analyses that they perform. However, such a merger would very likely reduce the modularity of the three-tiered approach to text processing that we have been following.

AN EXTENDED EXAMPLE

In this section, we illustrate in a more concrete fashion how the Unisys MUC-3 system goes about processing messages by examining in more detail what happens during the processing of a specific text, message TST1-MUC3-0099 in the MUC-3 corpus (see Figure 2). Our discussion will proceed through the various processing phases that have been identified.

Phase One: Message Pre-processing

In this phase, the message is parsed (by a special low-level processor) into its components and output in a form compatible with the KBIRD rule processing system. This processor is a special C program generated by MFPL, the ASL mentioned earlier in this paper. This phase produces text input of the following sort to the Prolog portion of the system, including default (header) information about the date and location.

TST1-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 2: Message TST1-MUC3-0099.

```
msg(id,"TST1-MUC3-0099").
msg(loc,"LIMA").
msg(date,[25,"OCT",89]).
msg(src,"EFE").
msg(type,'TEXT').
msg(text,['POLICE,'HAVE','REPORTED','THAT','TERRORISTS','TONIGHT', ...])
```

Phase Two: Keyword analysis

In the second phase, the keyword analysis component predicts three event classes—bombings with a probability of 87%, attacks with a probability of 66%, and murders with a probability of 63%. Figure 3 shows the particular words and word pairs which gave rise to these predicted event types. The last column in this table contains triples consisting of a probability, a word or two-word phrase, and its location in the text. Given our current thresholds, the murder prediction was judged to be too weak for further consideration.

Phase Three: KBIRD processing

KBIRD examines the text word by word and applies forward chaining rules whenever their pre-conditions are met. KBIRD's task is to take the event classes predicted by the keyword analysis stage and try to predict additional event classes as well as instantiate the predicted types with individual events. Event instances are associated with particular regions within the text. When an event instance is created, additional rules will be triggered to look for values to fill each of the instance's slots.

Sent	Type	Prob	Keys
2	bombing	55	[55,DAMAGE,20:21]
2	bombing	71	[57,THE,BOMBS,17:19] [71,DAMAGE,BUT,20:22]
3	bombing	79	[79,EXPLODED,27:29]
3	bombing	82	[82,EXPLODED,IN,27:29]
4	bombing	77	[54,THROWN,51:52] [77,PARKED,59:60]
4	bombing	84	[84,TWO,BOMBS,48:50]
5	bombing	80	[80,WINDOWS,88:89]
5	bombing	71	[57,THE,BOMBS,85:87] [71,WINDOWS,AND,88:90]
5	attack	66	[66,ATTACKS,WERE,77:99]
8	bombing	87	[56,BUS,167:168] [87,CALLAO,178:179]
8	bombing	75	[63,A,BUS,166:168] [75,PORT,OF,175:177]
10	murder	63	[63,KILLED,BY,217:219]
11	bombing	69	[69,DYNAMITE,231:232]

Figure 3: Keyword analysis predicts the likely occurrence of bombings and attacks in this message.

Predicting Additional Event Types. In some cases the co-occurrence of an instance of some event type predicted by the keyword-based analysis component with words or inferred concepts that have been detected in a text will allow KBIRD to infer additional event types. For example, the following KBIRD rule, which was triggered in the processing of message 0099, asserts that the occurrence of "BURNED" in the active voice in a message for which an instance of a bombing event has been discovered is enough to predict the likely occurrence of an arson event.

```
"BURNED" <. be_word(W),
actual_event('BOMBING',_,_),
[~predicted_event_type('ARSON')]
     ==> probable_event('ARSON').
```

Locating Events. The process of instantiating event types, or *locating events*, is initiated in KBIRD through a class of *locator rules*, which attempt to find "hot spots" in the text which seem to be discussing events of the predicted type. The following locator rules were used to detect bombing, attack, and arson instances in this message:

```
[probable_event('BOMBING')],
"BOMBED"@I <. ~be_word(W),
"BOMBED"@I ..> [potential_physical_target(_,_,_)],
{gen_event_id(ID)}
     ==> bombing(ID,'BOMBED'),
         syntax(ID,active).
```

Paraphrase: If the occurrence of a bombing event is likely and the word "bombed" occurs in the active voice (no preceding "be" word) with a potential physical target to its right in the same sentence, then infer an instance of a bombing event.

```
[probable_event('BOMBING')],
["THE"^"ATTACK*"]^be_word(W)^"CARRIED"^"OUT" .. [bomb_device],
{gen_event_id(ID)}
     ==> bombing(ID,'BE CARRIED OUT WITH BOMB'),
         syntax(ID,active).
```

Paraphrase: If the occurrence of a bombing event is likely and the phrase "The attack was carried out" occurs (or a variant with some other "be" word), and in the same sentence somewhere a bomb device is mentioned, then infer an instance of a bombing event.

```
[probable_event('ATTACK')],
["THE"^"ATTACK*"]^be_word(W)^"CARRIED"^"OUT" .. [~bomb_device],
{gen_event_id(ID)}
     ==> attack(ID,'BE CARRIED OUT'),
         syntax(ID,active).
```

Paraphrase: If the occurrence of an attack is likely and the phrase "The attack was carried out" occurs (or a variant with some other "be" word), and no mention is made of a bomb device in the same sentence, then infer an instance of an attack event.

```
[probable_event('ARSON')],
"BURNED" <. be_word(W) <.. [potential_physical_target(_,_,_)],
{gen_event_id(ID)}
     ==> arson(ID,'BE BURNED'),
         syntax(ID,passive).
```

Paraphrase: If the occurrence of an arson event is likely and the word "burned" occurs in the passive voice (with a "be" word to its left) with a mention of a potential physical target somewhere to the left in the same sentence, then infer an instance of an arson event.

Although the rule above for detecting an instance of an attack event will initially fire as the words in the message are examined sequentially by KBIRD and the phrase "THE ATTACK WAS CARRIED OUT" is encountered, the attack event instance that has been created will eventually be retracted when, in the same sentence, the description of a bomb device is encountered ("THE BOMBS"). On the other hand, the second rule for inferring instances of bombing events will suddenly have all of its antecedent constraints met when this latter phrase is encountered, and so it will fire to create a new instance of a bombing.

Locating perpetrator ids and orgs. The following two rules are triggered when, in the first sentence of 0099, the word "TERRORISTS" is encountered. The latter rule licenses the inference that "TERRORISTS" describes a potential perpetrator.

```
"TERRORISTS" ==> generic_perpetrator('TERRORIST').

generic_perpetrator(A)@P,
[~unlikely_perpetrator(Name)],
{get_full_text_at_loc(P,Name)}
     ==> potential_ind_perpetrator(A, Name).
```

Later, in the fourth paragraph of the text, the following rules are used to infer that the known guerrilla organizations "SHINING PATH" and "TUPAC AMARU REVOLUTIONARY MOVEMENT" have been encountered.

```
"SHINING"^"PATH" ==> organization('GUERRILLA').
"TUPAC"^"AMARU"^"REVOLUTIONARY"^"MOVEMENT"  ==> organization('GUERRILLA').
```

Locating a Physical Target. In processing the first three paragraphs of the text, a number of rules fire to trigger the recognition of potential physical targets. Embassies and vehicles are frequent physical targets, and so the following inference rules have been written to capture essential information about them:

```
"EMBASSIES"  ==> structure('DIPLOMAT OFFICE OR RESIDENCE','PLURAL').
"VEHICLE"    ==> vehicle(1).
"VEHICLES"   ==> vehicle('PLURAL').
```

```
vehicle(Q)   ==> structure('TRANSPORT VEHICLE',Q).

structure(Type,Quantity)@P, {get_full_text_at_loc(P,Text)}
  ==> structure(Text,Type,Quantity).

structure(Text,Type,Quantity)
  ==> potential_physical_target(Text,Type,Quantity).
```

Detecting Event Instances, Revisited. The discovery of a physical target satisfies the last of the antecedent constraints for the arson and the first bombing event locator rules mentioned earlier, and so actual events (event instances) can now be inferred by them. Actual events are represented in the chart as facts of the following sort:

```
chart(actual_event(BOMBING,E1,bombing),6:7).
chart(actual_event(BOMBING,E3,bombing),90:91).
chart(actual_event(BOMBING,E4,bombing),78:81).
chart(actual_event(BOMBING,E5,bombing),78:81).
chart(actual_event(BOMBING,E6,bombing),165:166).
chart(actual_event(ARSON,E7,arson),246:248).
```

Multiple bombing instances are created because of the many different ways in which the various rules for inferring bombing instances can be satisfied. It will be the job of the template generator to detect and merge references to the same event.

Generating Slot Values. Once an event instance has been asserted, KBIRD will begin to infer tmp clauses, which will later be written to a file to serve as input to the template generator for filling template slots. Each clause has as one of its parameters a score that indicates how likely it is to be an appropriate slot value. The following rules illustrate how a perpetrator that is a terrorist is favored in a bombing event.

```
actual_event(_,ID,bombing) .. potential_ind_perpetrator('TERRORIST',P)
==> tmp(ID, slot05, P, kbird, 95).

actual_event(_,ID,_) .. potential_ind_perpetrator(_,P)
==> tmp(ID, slot05, P, kbird, 50).
```

Similarly, the following rules illustrate how, in templates representing bombing events, organizations that have been identified as guerrilla groups are favored over drug cartels and military groups as likely values for the perpetrator ORG slot.

```
actual_event(_,ID,bombing)..organization( G, 'GUERRILLA')
  ==> tmp(ID, slot06, [G,'GUERRILLA'], kbird, 85).

actual_event(_,ID,bombing)..organization( G, 'DRUGGIES')
  ==> tmp(ID, slot06, [G,'REBELS'], kbird, 77).

actual_event(_,ID,bombing)..organization( G, 'MILITARY')
  ==> tmp(ID, slot06, [G,'MILITARY'], kbird, 35).
```

Phase Four: Template generator

Figure 4 contains the arson and bombing templates produced by the template generator for message TST1-MUC3-0099. Several bombing events were located, but they were all merged by the template generator into a single representation.

#	*Description*	*Filler used*	*Correct Filler (if different)*
0	message id	TST1-MUC3-0099	
1	template id	1	
2	date of incident	25 OCT 89	
3	type of incident	ARSON	
4	category of incident	TERRORIST ACT	
5	perpetrator: id of indiv	-	
6	perpetrator: id of org(s)	"SHINING PATH"	
7	perpetrator: confidence	CLAIMED OR ADMITTED: "SHINING PATH"	REPORTED AS FACT: "SHINING PATH"
8	physical target: id(s)	"BUSES"	
9	physical target: total num	10	
10	physical target: type(s)	TRANSPORT VEHICLE: "BUSES"	
11	human target: id(s)	-	
12	human target: total num	-	
13	human target: type(s)	-	
14	target: foreign nation(s)	-	
15	instrument: type(s)	*	
16	location of incident	PERU: LIMA (DEPARTMENT): LIMA (CITY)	
17	effect on physical target	SOME DAMAGE: "BUSES"	
18	effect on human target(s)	-	

#	*Description*	*Filler used*	*Correct Filler (if different)*
0	message id	TST1-MUC3-0099	
1	template id	2	
2	date of incident	01JUL89-31JUL89	24OCT89-25OCT89
3	type of incident	BOMBING	
4	category of incident	TERRORIST ACT	
5	perpetrator: id of indiv	"TERRORISTS"	
6	perpetrator: id of org(s)	"SHINING PATH"	← *This conjunct ok.* "TUPAC AMARU ..."
7	perpetrator: confidence	CLAIMED OR ADMITTED: "SHINING PATH"	REPORTED AS FACT: "TERRORISTS" POSSIBLE: "SHINING PATH" POSSIBLE: "TUPAC ..."
8	physical target: id(s)	"VEHICLES" "EMBASSIES"	"VEHICLE" ← *This conjunct ok.*
9	physical target: total num	PLURAL	
10	physical target: type(s)	TRANSPORT VEHICLE: "VEHICLES" DIPLOMAT OFFICE OR RESIDENCE: "EMBASSIES"	TRANSPORT VEHICLE: "VEHICLE" ← *This conjunct ok.*
11	human target: id(s)	-	
12	human target: total num	-	
13	human target: type(s)	-	
14	target: foreign nation(s)	-	USSR
15	instrument: type(s)	*	
16	location of incident	PERU: CALLAO (PORT)	PERU: LIMA (CITY): ORRANTIA (DISTRICT)
17	effect on physical target	DESTROYED: "EMBASSIES" DESTROYED: "VEHICLES"	SOME DAMAGE: EMBASSIES ← *This conjunct ok.*
18	effect on human target(s)	-	

Figure 4: Arson and bombing templates generated by the Unisys MUC-3 system for TST1-MUC3-0099.

The arson template generated by the system was almost completely correct. The only problem was that the perpetrator confidence reported for "SHINING PATH" was CLAIMED OR ADMITTED and not REPORTED AS FACT. In the bombing template generated by the system, the date was incorrectly identified as being a span of time in July instead of a span of time in October. The July inference was

based on information in the fifth paragraph. The system also failed to report the TUPAC AMARU group as a perpetrator ORG value, even though the group was identified in the text. An uninteresting bug in the template generator caused this error. Finally, rules for inferring that the physical targets belonged to foreign nations were not sensitive enough to be activated.

CONCLUSIONS

The value of the three-tiered approach realized in the Unisys MUC-3 system is two-fold. First, the terrorist domain is sufficiently well-defined that a deep linguistic analysis is often unneccessary, and using linguistic analysis sparingly provides a dramatic improvement in robustness and processing time. Second, in the MUC-3 evaluation task we have discovered that a small amount of modeling effort, i.e., writing KBIRD rules, produces a significant improvement in our ability to extract pertinent information. Since KBIRD is a forward chaining rule-driven methodology, the creation, modification and removal of rules is a very easy and intuitive process.

The three-tiered approach of combining traditional information retrieval and linguistic analysis techniques with the type of analysis that our knowledge-based information retrieval system, KBIRD, provides offers significant advantages to solving common text processing problems. The modularity of this approach allows us to utilize advances made in keyword analysis and NLP technology with relative ease.

REFERENCES

[1] Tim Finin, Rich Fritzson, and Dave Matuzsek. Adding forward chaining and truth maintenance to prolog. In *Fifth IEEE Conference on Artificial Intelligence Application*, pages 123–130, March 1989.

[2] Tim Finin, Robin McEntire, Carl Weir, and Barry Silk. A three-tiered approach to natural language text retrieval. In *Proceedings of the AAAI workshop on Natural Language Text Retrieval*, Los Angeles, July 1991.

[3] L. Hirschman, M. Palmer, J. Dowding, D. Dahl, M. Linebarger, R. Passonneau, F.-M. Lang, C. Ball, and C. Weir. The PUNDIT natural-language processing system. In *AI Systems in Government Conf.* Computer Society of the IEEE, March 1989.

[4] Chris Paice. Another stemmer. *SIGIR Forum*, Fall 1990.

[5] Fernando C.N. Pereira and David H.D. Warren. Definite clause grammars for language analysis—a survey of the formalism and a comparison with augmented transition networks. *Artificial Intelligence*, 13(3):231–278, 1980.

[6] Bob Pollack. Message format processing language. Manual, Unisys Center for Advanced Information Technology, August 1989. Version 2.1.

[7] Carl Weir, Tim Finin, Barry Silk, Marcia Linebarger, and Robin McEntire. Knowledge-based strategies for robust text-understanding. The Eighth Annual Intelligence Community AI/Advance Computing Symposium, March 1991.