SRI International

FASTUS: A System for Extracting Information from Natural-Language Text

Technical Note No. 519

November 19, 1992

By: Jerry R. Hobbs, Program Director, Natural Language
Douglas E. Appelt, Sr. Computer Scientist
John S. Bear, Computer Scientist
David J. Israel, Sr. Computer Scientist
W. Mabry Tyson, Manager, Computer Facility

Artificial Intelligence Center
Computing and Engineering Sciences Division

This work was supported in part by the Defense Advanced Research Projects Agency under Office of Naval Research contract N00014-90-C-0220, and by an internal research and development grant from SRI International.

333 Ravenswood Avenue • Menlo Park, CA 94025-3493 • (415) 326-6200 • FAX: (415) 326-5512 • Telex: 334486

FASTUS: A System for Extracting Information from Natural-Language Text

Jerry R. Hobbs, Douglas Appelt, John Bear,
David Israel, and Mabry Tyson
Artificial Intelligence Center
SRI International
Menlo Park, California

Abstract

FASTUS is a system for extracting information from free text in English, and potentially other languages as well, for entry into a database, and potentially for other applications. It works essentially as a cascaded, nondeterministic finite state automaton. There are four steps in the operation of FASTUS. In Step 1 sentences are scanned for certain trigger words to determine whether further processing should be done. In Step 2 noun groups, verb groups, and prepositions and some other particles are recognized. The input to Step 3 is the sequence of phrases recognized in Step 2; patterns of interest are identified in Step 3 and corresponding "incident structures" are built up. In Step 4 incident structures that derive from the same incident are identified and merged, and these are used in generating database entries. FASTUS is an order of magnitude faster than any comparable system; it can process a news report in an average of less than eleven seconds. This translates directly into fast development time. In the three and a half weeks between its first use and the MUC-4 evaluation in May 1992, we were able to build up its domain knowledge to a point where it was among the leaders in the evaluation.

1 Introduction

FASTUS is a (slightly permuted) acronym for Finite State Automaton Text Understanding System. It is a system for extracting information from free text in English, and potentially other languages as well, for entry into a database, and potentially for other applications. It works essentially as a set of cascaded, nondeterministic finite state automata.

In Section 2 we describe the MUC-4 evaluation, for which the system was initially built. In Section 3 we discuss the important distinction between information extraction systems and text understanding systems. In Section 4 we review previous finite-state approaches to natural language processing. Section 5 describes the overall architecture of the FASTUS system, and Sections 6 through 9 describe the individual components. A number of parameter settings are possible in FASTUS and Section 10 describes our experimentation with these parameters. Section 11 describes the speed of the system, and Section 12 gives the history of its development. Section 13 describes the results for the FASTUS system in the MUC-4 evaluation, and Section 14 gives a brief error analysis. Section 15 discusses future directions, and Section 16 summarizes the advantages of the FASTUS approach and the lessons learned from using it in the MUC-4 evaluation.

2 The MUC-4 Evaluation

SRI International participated in the recent MUC-4 evaluation of text-understanding systems (Sundheim, 1992), the fourth in a series of evaluations. The methodology chosen for this evaluation was to score a system's ability to fill in slots in templates summarizing the content of newspaper articles on Latin American terrorism. The articles ranged from one third of a page to two pages in length. The template-filling task required identifying, among other things, the perpetrators and victims of each terrorist act described in an article, the occupations of the victims, the type of physical entity attacked or destroyed, the date, the location, and the effect on the targets. Many articles described multiple incidents, while other texts were completely irrelevant.

The following are some relevant excerpts from a sample terrorist report (TST2-MUC4-0048), the full text of which is in Appendix I. Most of the examples in this paper come from this message; it was selected by the organizers of the MUC-4 evaluation as exhibiting many relevant problems.

> San Salvador, 19 Apr 89 (ACAN-EFE) – [TEXT] Salvadoran President-elect Alfredo Cristiani condemned the terrorist killing of Attorney General Roberto Garcia Alvarado and accused the Farabundo Marti National Liberation Front (FMLN) of the crime.
>
> ...

> Garcia Alvarado, 56, was killed when a bomb placed by urban guerrillas on his vehicle exploded as it came to a halt at an intersection in downtown San Salvador.
>
> ...
>
> Vice President-elect Francisco Merino said that when the attorney general's car stopped at a light on a street in downtown San Salvador, an individual placed a bomb on the roof of the armored vehicle.
>
> ...
>
> According to the police and Garcia Alvarado's driver, who escaped unscathed, the attorney general was traveling with two bodyguards. One of them was injured.

This text is from the TST2 set of one hundred messages used in the final evaluation of MUC-3 in May 1991.

Some of the corresponding database entries are as follows:

Incident: Date	- 19 Apr 89
Incident: Location	El Salvador: San Salvador (city)
Incident: Type	Bombing
Perpetrator: Individual ID	"urban guerrillas"
Perpetrator: Organization ID	"FMLN"
Perpetrator: Organization Confidence	Suspected or Accused by Authorities: "FMLN"
Physical Target: Description	"vehicle"
Physical Target: Effect	Some Damage: "vehicle"
Human Target: Name	"Roberto Garcia Alvarado"
Human Target: Description	"attorney general": "Roberto Garcia Alvarado" "driver" "bodyguards"
Human Target: Effect	Death: "Roberto Garcia Alvarado" No Injury: "driver" Injury: "bodyguards"

The complete templates for this text are in Appendix II.

The seventeen sites participating in MUC-4 had available a development corpus of 1500 texts, together with their corresponding templates and an

automatic scoring program. The same corpus was used in the MUC-3 evaluation (Sundheim, 1991). In the last week of May 1992 the systems were tested on two new sets of 100 messages each, one from the same time slice as the development messages (TST3) and one from a new time slice (TST4). The results were reported at a workshop at Tyson's Corner, Virginia, in June 1992.

The principal measures in the MUC-4 evaluation were recall and precision. *Recall* is the number of answers the system got right divided by the number of possible right answers. It measures how comprehensive the system is in its extraction of relevant information. *Precision* is the number of answers the system got right divided by the number of answers the system gave. It measures the system's accuracy. For example, if there are 100 possible answers and the system gives 80 answers and gets 60 of them right, its recall is 60% and its precision is 75%.

In addition, a combined measure, called the F-score, was used. The F-score is defined as follows:

$$F = \frac{(\beta^2+1)PR}{\beta^2 P+R}$$

where P is precision, R is recall, and β is a parameter encoding the relative importance of recall and precision. If $\beta = 1$, they are weighted equally. If $\beta > 1$, precision is more significant; if $\beta < 1$, recall is. Official scores were computed for $\beta = .5$, $\beta = 1$, and $\beta = 2$.

3 Two Types of System

One can distinguish between two types of natural language systems: *information extraction* systems and *text understanding* systems. In information extraction,

- Only a fraction of the text is relevant; in the case of the MUC-4 terrorist reports, probably only about 10% of the text is relevant.
- Information is mapped into a predefined, relatively simple, rigid target representation; this condition holds whenever entry of information into a database is the task.
- The subtle nuances of meaning and the writer's goals in writing the text are of no interest.

This contrasts with text understanding, where

- The aim is to make sense of the entire text.
- The target representation must accommodate the full complexities of language.
- One wants to recognize the nuances of meaning and the writer's goals.

The task in the MUC evaluations is information extraction, not text understanding. When SRI participated in the MUC-3 evaluation in 1991, we used TACITUS, a text-understanding system (Hobbs et al., 1992a; Hobbs et al., 1992b). Using it for the information extraction task gave us a high precision, the highest of any of the sites. However, because it was spending so much of its time attempting to make sense of portions of the text that were irrelevant to the task, the system was extremely slow. As a result, development time was slow, and consequently our recall was mediocre.

FASTUS, by contrast, is an information extraction system, rather than a text understanding system. Our motivation in developing FASTUS was to build a system that was more appropriate to the information extraction task.

4 The Finite-State Approach

The inspiration for FASTUS was threefold. First, we were struck by the strong performance that the group at the University of Massachusetts got out of a fairly simple system (Lehnert et al., 1991). It was clear they were not doing anything like the depth of preprocessing, syntactic analysis, or pragmatics that was being done by the systems at SRI, General Electric, or New York University. They were not doing a lot of processing. But they were doing the *right* processing.

The second source of inspiration was Pereira's work on finite-state approximations of grammars (Pereira, 1990), especially the speed of the implemented system.

Speed was the third source. It was simply too embarassing to have to report at the MUC-3 conference that it took TACITUS 36 hours to process 100 messages. FASTUS has brought that time down to less than 12 minutes.

Finite-state models are clearly not adequate for full natural language processing. However, if context-free parsing cannot be cost-effectively applied to real-world text, then an efficient text processor might make use of weaker language models, such as regular or finite-state grammars. Every

computational linguistics graduate student knows, from the first textbook that introduces the Chomsky hierarchy, that English has constructs, such as center embedding, that cannot described by any finite-state grammar. This fact has biased researchers away from serious consideration of possible applications of finite-state grammars to difficult problems.

Church (1980) was the first to advocate finite-state grammars as a processing model for language understanding. He contended that, although English is clearly not a regular language, memory limitations make it impossible for people to exploit that context-freeness in its full generality, and therefore a finite-state mechanism might be adequate in practice as a model of human linguistic performance. A computational realization of memory limitation as a depth cutoff was implemented by Black (1989).

More recently, Pereira and Wright (1991) have developed methods for constructing finite-state grammars from context free grammars that overgenerate in certain systematic ways. The finite-state grammar could be applied in situations, for example, as the language model in a speech understanding system, where computational considerations are paramount.

At this point, the limitations of the application of finite-state grammars to natural-language processing have not yet been determined. We believe our research establishes that these simple mechanisms can achieve a lot more than has previously been thought possible.

5 Overview of the FASTUS Architecture

The input text is first preprocessed to ensure that the text is in a standardized format for the remainder of the processing. Spelling correction is done at this point as well. The preprocessed text is then given to the FASTUS system proper.

The operation of FASTUS is comprised of four steps:

1. Triggering
2. Recognizing Phrases
3. Recognizing Patterns
4. Merging Incidents

These steps are described in the next four sections. A postprocessing phase then converts the incident structures generated by FASTUS into the format required for the MUC-4 templates.

The system is implemented in CommonLisp and runs on both Sun and Symbolics machines.

6 Triggering

In the first pass over a sentence, trigger words are searched for. There is at least one trigger word for each pattern of interest that has been defined. Generally, these are the least frequent words required by the pattern. For example, in the pattern

take <HumanTarget> hostage

"hostage" rather than "take" is the trigger word. There are at present 253 trigger words.

In addition, the names of people identified in previous sentences as victims are also treated, for the remainder of the text, as trigger words. This allows us, for example, to pick up occupations of victims when they occur in sentences with no other triggers, as in

> Hector Oqueli and Gilda Flores were assassinated yesterday.
>
> Gilda Flores was a member of the Democratic Socialist Party (PSD) of Guatemala.

Finally, on this pass, full names are searched for, so that subsequent references to surnames can be linked to the corresponding full names. Thus, if one sentence refers to "Ricardo Alfonso Castellar" but does not mention his kidnapping, while the next sentence mentions the kidnapping but only uses his surname, we can enter Castellar's full name into the template.

The performance of FASTUS on Message 48 of TST2 is illustrative of its performance in general. In that message, 21 of 30 sentences were triggered. 13 of the 21 triggered sentences were relevant. There is very little penalty for passing irrelevant sentences on to further processing since the system is so fast, especially on irrelevant sentences.

Eight of the nine nontriggered sentences were irrelevant. The one relevant, nontriggered sentence was

> There were seven children, including four of the vice president's children, in the home at the time.

It does not help to recognize this sentence as relevant as we do not have a pattern that would match it.

The missing pattern is

<HumanTarget> be in <PhysicalTarget>

which would pick up human targets who were in known physical targets. In order to have this sentence triggered, we would have to take the head nouns of known physical targets to be temporary triggers for the remainder of the text, as we do with named human targets.

7 Recognizing Phrases

The problem of syntactic ambiguity is AI-complete. That is, we will not have systems that reliably parse English sentences correctly until we have encoded much of the real-world knowledge that people bring to bear in their language comprehension. For example, noun phrases cannot be reliably identified because of the prepositional phrase attachment problem. However, certain syntactic constructs can be reliably identified. One of these is the noun group, that is, the head noun of a noun phrase together with its determiners and other left modifiers. Another is what we are calling the "verb group", that is, the verb together with its auxiliaries and any intervening adverbs. Moreover, an analysis that identifies these elements gives us exactly the units we most need for recognizing patterns of interest.

Pass Two in FASTUS identifies noun groups, verb groups, and several critical word classes, including prepositions, conjunctions, relative pronouns, and the words "ago" and "that". Phrases that are subsumed by larger phrases are discarded. Overlapping phrases are rare, but where they occur they are kept. This sometimes compensates for an incorrect analysis in Pass Two.

Noun groups are recognized by a 37-state nondeterministic finite state automaton. This encompasses most of the complexity that can occur in English noun groups, including numbers, numerical modifiers like "approximately", other quantifiers and determiners, participles in adjectival position, comparative and superlative adjectives, conjoined adjectives, and arbitrary orderings and conjunctions of prenominal nouns and noun-like adjectives. Thus, among the noun groups recognized are

approximately 5 kg
more than 30 peasants
the newly elected president
the largest leftist political force
a government and military reaction

Verb groups are recognized by an 18-state nondeterministic finite state machine. They are tagged as Active, Passive, Gerund, and Infinitive. Verbs are sometimes locally ambiguous between active and passive senses, as the verb "kidnapped" in the two sentences,

> Several men kidnapped the mayor today.
> Several men kidnapped yesterday were released today.

These are tagged as Active/Passive, and Pass Three resolves the ambiguity if necessary.

Certain relevant predicate adjectives, such as "dead" and "responsible", are recognized, as are certain adverbs, such as "apparently" in "apparently by". However, most adverbs and predicate adjectives and many other classes of words are ignored altogether. Unknown words are ignored unless they occur in a context that could indicate they are surnames.

The complete specifications for noun groups and verb groups are given in Appendix III.

Lexical information is read at compile time, and a hash table associating words with their transitions in the finite-state machines is constructed. There is a hash table entry for every morphological variant of a word. The TACITUS lexicon of 20,000 words is used for lexical information. Morphological expansion of these words results in 43,000 morphological variants in the hash table. During the actual running of the system on the texts, only the state transitions accessed through the hash table are seen.

The output of the second pass for the first sentence of Message 48 of TST2 is as follows:

Noun Group:	Salvadoran President-elect
Name:	Alfredo Cristiani
Verb Group:	condemned
Noun Group:	the terrorist
Verb Group:	killing
Preposition:	of
Noun Group:	Attorney General
Name:	Roberto Garcia Alvarado
Conjunction:	and
Verb Group:	accused

Noun Group:	the Farabundo Marti National Liberation Front (FMLN)
Preposition:	of
Noun Group:	the crime

The verb groups "condemned" and "accused" are labeled "Active/Passive". The word "killing" which was incorrectly identified as a verb group is labeled as a Gerund. This mistake is common enough that we have implemented patterns to get around it in Pass Three.

On Message 48 of TST2, 243 of 252 phrases, or 96.4%, were correctly recognized. Of the 9 mistakes, 5 were due to nouns being misidentified as verbs or verbs as nouns. the other 4 mistakes were due to simple bugs of the type that frequently creep into code during development.

We implemented and considered using a part-of-speech tagger to help in Pass Two, but there was no clear improvement and it would have doubled the time the system took to process a message.

8 Recognizing Patterns

The input to Pass Three of FASTUS is a list of phrases in the order in which they occur. Anything that is not included in a phrase in the second pass is ignored in the third pass. Patterns of interest are encoded as finite state machines, where state transitions are effected by phrases. The state transitions are driven off the head words in the phrases. That is, a set of state transitions is associated with each relevant head word-phrase type pair, such as "mayor-NounGroup", "kidnapped-PassiveVerbGroup", "killing-NounGroup", and "killing-GerundVerbGroup". In addition, some nonhead words can trigger state transitions. For example, "bomb blast" is recognized as a bombing.

We implemented 95 patterns for the MUC-4 application. Among the patterns are the following ones that are relevant to Message 48 of TST2:

killing of <HumanTarget>
<GovtOfficial> accused <PerpOrg>
bomb was placed by <Perp> on <PhysicalTarget>
<Perp> attacked <HumanTarget>'s <PhysicalTarget> with <Device>
<HumanTarget> was injured
<HumanTarget>'s body

As patterns are recognized, incident structures are built up. For example, the sentence

> Guerrillas attacked Merino's home in San Salvador 5 days ago with explosives.

matches the pattern

> <Perp> attacked <HumanTarget>'s <PhysicalTarget> in <Location> <Date> with <Device>

This causes the following incident to be constructed.

Incident:	ATTACK/BOMBING
Date:	14 Apr 89
Location:	El Salvador: San Salvador
Instr:	"explosives"
Perp:	"guerrillas"
PTarg:	"Merino's home"
HTarg:	"Merino"

The incident type is an attack or a bombing, depending on the Device. There was a bug in this pattern that caused the system to miss picking up the explosives as the instrument. In addition, it is disputable whether Merino should be listed as a human target. In the official key template for this message, he is not. But it seems to us that if someone's home is attacked, it is an attack on him.

A certain amount of "pseudo-syntax" is done while patterns are being recognized. In the first place, the material between the end of the subject noun group and the beginning of the main verb group must be read over. There are patterns to accomplish this. Two of them are as follows:

> Subject {Preposition NounGroup}* VerbGroup
>
> Subject Relpro {NounGroup | Other}* VerbGroup {NounGroup | Other}* VerbGroup

The first of these patterns reads over prepositional phrases. The second over relative clauses. The verb group at the end of these patterns takes the subject noun group as its subject. There is another pattern for capturing the content encoded in relative clauses:

Subject Relpro {NounGroup | Other}* VerbGroup

Since the finite-state mechanism is nondeterministic, the full content can be extracted from the sentence

The mayor, who was kidnapped yesterday, was found dead today.

One branch discovers the incident encoded in the relative clause. Another branch marks time through the relative clause and then discovers the incident in the main clause. These incidents are then merged.

A similar device is used for conjoined verb phrases. The pattern

Subject VerbGroup {NounGroup | Other}* Conjunction VerbGroup

allows the machine to nondeterministically skip over the first conjunct and associate the subject with the verb group in the second conjunct. Thus, in the sentence

Salvadoran President-elect Alfredo Cristiani condemned the terrorist killing of Attorney General Roberto Garcia Alvarado and accused the Farabundo Marti National Liberation Front (FMLN) of the crime.

one branch will recognize the killing of Garcia and another the fact that Cristiani accused the FMLN.

The second sort of "pseudo-syntax" that is done while recognizing patterns is attaching genitives, "of" complements, and appositives to their heads, and recognizing noun group conjunctions. Thus, in

seven children, including four of the vice-president's children

the genitive "vice-president's" will be attached to "children". The "of" complement will be attached to "four", and since "including" is treated as a conjunction, the entire phrase will be recognized as conjoined noun groups. (Note that the official key for MUC-4 lists "seven children" and "four of the vice-president's children" as separate entries. It was not part of the task to discover complex set inclusion relations.)

In Message 48 of TST2, there were 18 relevant patterns. FASTUS recognized 12 of them completely. Because of bugs in implemented patterns, 3 more patterns were recognized only partially. One implemented pattern failed completely because of a bug. Specifically, in the sentence

A niece of Merino's was injured.

the genitive marker took the system into a state in which it was not expecting a verb group.

Two more patterns were missing entirely. The pattern

<HumanTarget1> <VerbGroup> with <HumanTarget2>

would have matched

... the attorney general was traveling with two bodyguards.

and consequently would have recognized the two bodyguards as human targets along with the attorney general.

The second pattern is

<HumanTarget> be in <PhysicalTarget>

mentioned above.

A rudimentary sort of pronoun resolution is done by FASTUS. If (and only if) a pronoun appears in a Human Target slot, an antecedent is sought. First the noun groups of the current sentence are searched from left to right, up to four phrases before the pronoun. Then the previous sentences are searched similarly for an acceptable noun group in a left-to-right fashion, the most recent sentence first. This is continued until a paragraph break is encountered, and if nothing is found by then, the system gives up. A noun group is an acceptable antecedent if it is a possible human target and agrees with the pronoun in number. This algorithm worked in 100% of the relevant cases in the first 200 messages of the development set. However, in its one application in Message 48 of TST2, it failed. The example is

> According to the police and Garcia Alvarado's driver, who escaped unscathed, the attorney general was traveling with two bodyguards. One of them was injured.

The algorithm incorrectly identifies "them" as "the police".

9 Merging Incidents

As incidents are found they are merged with other incidents found in the same sentence. Those remaining at the end of the processing of the sentence are then merged, if possible, with the incidents found in previous sentences.

For example, in the first sentence of Message 48 of TST2, the incident

Incident:	KILLING
Perp:	–
Confid:	–
HTarg:	"Roberto Garcia Alvarado"

is generated from the phrase

killing of Attorney General Roberto Garcia Alvarado

while the incident

Incident:	INCIDENT
Perp:	FMLN
Confid:	Suspected or Accused by Authorities
HTarg:	–

is generated from the clause

Salvadoran President-elect Alfredo Cristiani ... accused the Farabundo Marti National Liberation Front (FMLN)

These two incidents are merged, by merging the KILLING and the INCIDENT into a KILLING, and by taking the union of the other slots.

Incident:	KILLING
Perp:	FMLN
Confid:	Suspected or Accused by Authorities
HTarg:	"Roberto Garcia Alvarado"

Merging is blocked if the incidents have incompatible types, such as a KIDNAPPING and a BOMBING. It is also blocked if they have incompatible dates or locations.

There are fairly elaborate rules for merging the noun groups that appear in the Perpetrator, Physical Target, and Human Target slots. A name can be merged with a description, as "Garcia" with "attorney general", provided the description is consistent with the other descriptions for that name. A precise description can be merged with a vague description, such as "person", with the precise description as the result. Two precise descriptions can be merged if they are semantically compatible. The descriptions "priest" and

"Jesuit" are compatible, while "priest" and "peasant" are not. When precise descriptions are merged, the longest string is taken as the result. If merging is impossible, both noun groups are listed in the slot.

We experimented with a further heuristic for merging incidents. If the incidents include named human targets, we do not merge them unless there is an overlap in the names. This heuristic results in about a 1% increase in recall. In Message 48 of TST2, the heuristic prevents the Bombing of Garcia Alvarado's car from being merged with the Bombing of Merino's home.

There were 13 merges altogether in processing Message 48 of TST2. Of these, 11 were valid.

One of the two bad merges was particularly unfortunate. The phrase

... Garcia Alvarado's driver, who escaped unscathed, ...

correctly generated an attack incident with no injury to the human target, the driver:

Incident:	ATTACK
Perp:	–
PTarg:	–
HTarg:	"Garcia Alvarado's driver"
HEffect:	No Injury

This was merged with the attack on Merino's home

Incident:	BOMBING
Perp:	"guerrillas"
PTarg:	"Merino's home"
HTarg:	"Merino"
HEffect:	–

to yield the combined incident

Incident:	BOMBING
Perp:	"guerrillas"
PTarg:	"Merino's home"
HTarg:	"Merino": "Garcia Alvarado's driver"
HEffect:	No Injury

That is, it was assumed that Merino was the driver. The reason for this mistake was that while a certain amount of consistency checking is done before merging victims, and while the system knows that drivers and vice presidents-elect are disjoint sets, the fact that Merino was the vice president-elect was recorded only in a table of titles, and consistency checking did not consult that table.

10 Controlling the FASTUS System

In the course of designing the system, we parameterized a number of characteristics of the system's operation because we believed that the parameterized behavior would reflect tradeoffs in recall versus precision. Subsequent testing revealed that many of these parameters result in both higher recall *and* higher precision when in one state or the other, and therefore we left them permanently in their most advantageous state. Those parameters that seemed to affect recall at the expense of precision were set to produce an optional test run in which we attempted to maximize the system's recall. The effect of these parameters could be described in general as distrusting the system's filters' ability to eliminate templates for incidents that were defined by the MUC-4 rules as being of no interest, including military incidents, incidents in uninteresting countries, and incidents that occurred more than two months before the date of the article. We observed a small but measurable increase in recall at the expense of precision by distrusting our filters.

The following parameters were implemented and tested on 300 texts before we decided on the settings for the final run of the MUC-4 evaluation. Most of them depend on the specific rules of the MUC-4 evaluation.

- *Conservative Merging.* A major emphasis of MUC-4, largely because of its scoring algorithm, was the proper individuation of incidents. When the Conservative Merging option is selected in FASTUS, the system would not merge incidents that had nonoverlapping targets with proper names. When not selected, any merges consistent with the incident types were permitted. Testing revealed that merging should *always* be conservative.

- *Civilian Target Requirement.* Incidents that involved only the military were of no interest in MUC-4. The Civilian Target Requirement

filter would reject any template that did not have at least one non-military target, including templates that identified a perpetrator, but no physical or human target at all. This option appears to produce a recall-precision tradeoff of about one or two points. That is, recall improved at the expense of precision if we distrusted our system and assumed that there really were civilian targets but that they were missed by the system.

- *Subjectless Verb Groups.* This parameter would allow the system to generate an incident structure from a verb together with its object, even if its subject could not be determined. Although early tests showed a recall-precision tradeoff, subsequent and more thorough testing indicated that this should always be done.

- *Filter Many-Target Templates.* This filter would disallow any template that had more than 100 targets, on the supposition that such templates often result from vague or general descriptions, and hence would be irrelevant to MUC-4. This turns out to be a correct heuristic, but only if the number of targets is evenly divisible by 100. (An airline bombing with 307 victims is certainly interesting, while "70,000 peasants have been killed" is probably vague).

- *Military Filtering.* This heuristic causes the system to eliminate all military targets from templates, on the belief that we may have incorrectly merged a military incident with a civilian incident and incorrectly reported the union of the two. Tests show that this filtering improves precision slightly.

- *Liberal Perpetrator Org.* Setting this parameter would cause the system to pick any likely perpetrator organization out of the text, ignoring whatever the text actually says. Testing showed that this parameter had no effect, which was such a surprising result that we distrust it, and regard our testing as inconclusive.

- *Spelling Correction.* This parameter controls how much spelling correction the system does. Our experiments indicated that spelling correction hurts, primarily because novel proper names get corrected to other words, and hence are lost. We tried a weaker version of spelling correction which would correct only misspelled words that did not occur on a large list of proper names that we had assembled. This showed

an improvement, but spelling correction still had a small negative effect. This was also a surprising result, and we were not willing to abandon spelling correction, and ran all tests with weak spelling correction enabled, although to some extent a complete lack of spelling correction is compensated for by the presence of common misspellings of important domain words like "guerrilla" and "assassinate" in the lexicon.

- *Stale Date Filtering.* This parameter causes filtering of any template that has a date that is earlier than two months before the date of the article. Eliminating this filtering produces an increase in recall at the expense of precision, the magnitude of which depends on how well our date detection currently works. We would expect about a one-point tradeoff.

- *Weak Location Filtering.* Normally, if the system's location detection routine finds that the location of an incident is impossible according to the system's location database, it eliminates the template. If this flag is set, however, the template will be produced using only the country as the location. Testing shows that this is always desirable.

11 How Fast is FASTUS?

The system is extremely fast. The entire TST3 set of 100 messages, ranging from a third of a page to two pages in length, required 11.8 minutes of CPU time on a Sun SPARC-2 processor. The elapsed real time was 15.9 minutes, but observed time depends on the particular hardware configuration involved.

In more concrete terms, this means that FASTUS can read 2,375 words per minute. It can analyze one text in an average of 9.6 seconds. This translates into 9,000 texts per day.

There is an old joke that goes like this:

> Q: What is the difference between computer science and artificial intelligence?
>
> A: In computer science you write programs to do quickly what people do slowly. In artificial intelligence, it is just the opposite.

According to this definition, FASTUS is computer science.

This fast run time translates directly into fast development time, as described in the next section.

12 Development History

FASTUS was originally conceived, in December 1991, as a preprocessor for TACITUS that could also be run in a stand-alone mode. It was only in the middle of May 1992, considerably later in our development, that we decided the performance of FASTUS on the MUC-4 task was so high that we could make FASTUS our complete system.

Most of the design work for the FASTUS system took place during January. The ideas were tested out on finding incident locations and proper names in February. With some initial favorable results in hand, we proceeded with the implementation of the system in March. The implementation of Pass Two was completed in March, and the general outline of Pass Three was completed by the end of April. On May 6, we did the first test of the FASTUS system on TST2, which had been withheld as a fair test, and we obtained a score of 8% recall and 42% precision. At that point we began a fairly intensive effort to hill-climb on all 1300 development texts, doing periodic runs on the fair test to monitor our progress. This effort culminated in a score of 44% recall and 57% precision in the wee hours of June 1, when we decided to run the official test. As the chart in Figure 1 makes clear, the rate of progress was rapid enough that even a few hours of work could be shown to have a noticeable impact on the score. Our scarcest resource was time, and our supply of it was eventually exhausted well before the point of diminishing returns.

We were thus able, in three and a half weeks, to increase the system's F-score by 36.2 points, from 13.5 to 49.7.

During the course of development, the overall run time for 100 messages increased approximately 50%, but we attribute this increase to the decision to treat more sentences as relevant. It appears possible to increase the coverage of the system without an unacceptable increase in processing time.

13 Results in the MUC-4 Evaluation

On TST3, we achieved a recall of 44% with precision of 55% in the all-templates row, for an F-score ($\beta = 1$) of 48.9. On TST4, the test on incidents

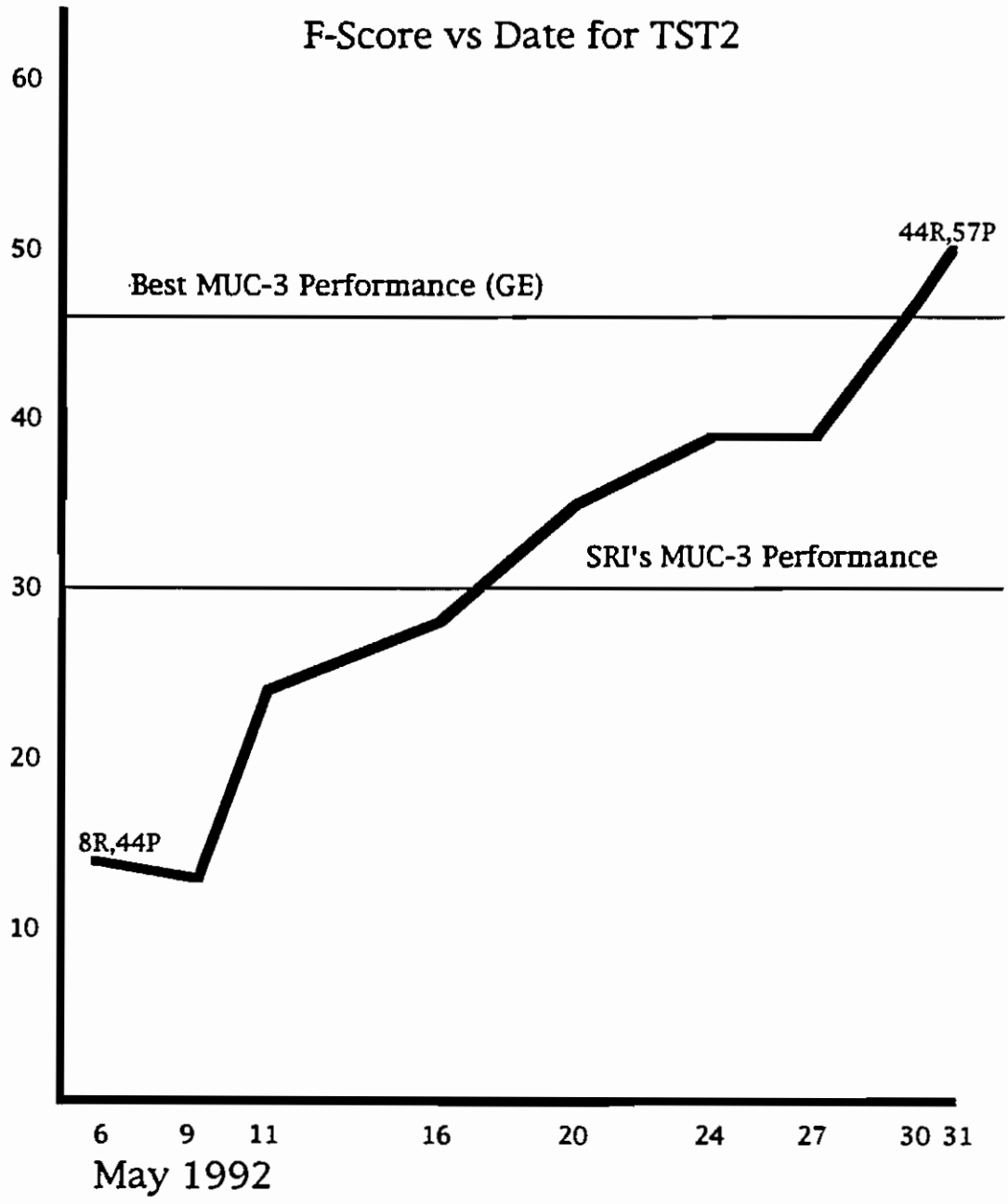

Figure 1: Plot of F-Score versus Date for FASTUS Development

from a different time span, we observed, surprisingly, an identical recall score of 44%; however our precision fell to 52%, for an F-score of 47.7. It was reassuring to see that there was very little degradation in performance when moving to a time period over which the system had not been trained.

We also submitted a run in which we attempted to maximize the system's recall by not filtering military targets, and allowing incidents with stale dates. On TST3, this led to a two-point increase in recall at the expense of one point in precision. On TST4, our recall did not increase, although our precision fell by a point, giving us a lower F-score on this run. These results were consistent with our observations during testing, although our failure to produce even a small increase in recall on TST4 was somewhat disappointing.

Only General Electric's system performed significantly better (a recall of 62% and a precision of 53% on TST3), and their system has been under development for over five years (Sundheim, 1992). Given the slope of our development curve in Figure 1, we believe we could have achieved scores in that range with another month or two of effort. It is unlikely that human coders would achieve an agreement of more than around 80% on this task. Thus, we believe this technology can perform 75% as well as humans, and considerably faster. Therefore, combining this technology with a good user interface can significantly increase productivity on information extraction tasks.

14 Error Analysis

FASTUS made 25 errors on Message 48 of TST2, where a wrong answer, a missing answer, and a spurious answer are all counted as errors. (There is in principle no limit to the number of possible errors, since arbitrarily many spurious entries could be given. However, practically the number of possible errors is around 80. If no entries at all are made in the templates, that counts as 55 errors. If all the entries are made and are correct, but combined into a single template, that counts as 48 errors---the 24 missing entries in the smaller template and the 24 spurious entries in the larger.)

The sources of the errors are as follows:

Missing Patterns (2)	9
Bad Merges (2 of 13)	7
Military "armored car" Filtered Out	4
Answer Disputable	3
Bug in Existing Pattern	2
Bad Pronoun Resolution	1
Mysterious	1

Because of the missing patterns, we failed to find the children and the bodyguards as human targets. The bad merges resulted in the driver being put into the wrong template. The armored car was found as a physical target in the attack against Garcia Alvarado, but armored cars are viewed as military, and military targets are filtered out just before the templates are generated. The disputable answer is Merino as a human target in the bombing of his home.

We do not know to what extent this pattern of causes of errors is representative of the performance of the system on the corpus as a whole.

15 Future Directions

If we had had one more month to work on the MUC-4 task, we would have spent the first week developing a rudimentary pattern specification language. We are now developing a language that will enable a novice user to begin to specify patterns in a new domain within hours of being introduced to the system. The pattern specification language will allow the user to define structures, to specify patterns either graphically or in regular expressions augmented by assignments to fields of the structures, and to define a sort hierarchy to control the merging of structures.

We would also like to apply the system to a new domain. Our experience with the MUC-4 task leads us to believe we could achieve reasonable performance on a new domain within two months.

Finally, it would be interesting to convert FASTUS to a new language. There is not much linguistic knowledge built into the system. What there is probably amounted to no more than two weeks' coding. For this reason, we believe it would require no more than one or two months to convert the system to another language. This is true even for a language as seemingly

dissimilar to English as Japanese. In fact, our approach to recognizing phrases was inspired in part by the bunsetsu analysis of Japanese.

16 Conclusions

FASTUS was more successful than we ever dreamed when the idea was originally conceived. In retrospect, we attribute its success to the fact that its processing is extremely well suited to the demands of the task. The system's Pass Three works successfully because the input from Pass Two is already reliably processed. Pass Two does only the linguistic processing that can be done reliably and fast, ignoring all the problems of making attachment decisions, and the ambiguity introduced by coordination and appositives. This input is adequate for Pass Three because the domain pragmatics is sufficiently constrained that, given this initial chunking, the relevant information can be reliably detected and extracted.

The advantages of the FASTUS system are as follows:

- It is conceptually simple. It is a set of cascaded finite-state automata.
- The basic system is relatively small, although the dictionary and other lists are potentially very large.
- It is effective. Only General Electric's system performed significantly better than FASTUS, and it has been under development for a number of years.
- It has very fast run time. The average time for analyzing one message is less than 10 seconds. This is nearly an order of magnitude faster than comparable systems.
- In part because of the fast run time, it has a very fast development time. This is also true because the system provides a very direct link between the texts being analyzed and the data being extracted.

FASTUS is not a text understanding system. It is an information extraction system. But for information extraction tasks, it is perhaps the most convenient and most effective system that has been developed.

One of the lessons to be learned from our FASTUS experience is that a MUC-like task is much easier than anyone ever thought. Although the full linguistic complexity of the MUC texts is very high, with long sentences and interesting discourse structure problems, the relative simplicity of the

information-extraction task allows much of this linguistic complexity to be bypassed—indeed much more than we had originally believed was possible. The key to the whole problem, as we see it from our FASTUS experience, is to do exactly the right amount of syntax, so that pragmatics can take over its share of the load. For the MUC task, we think FASTUS displays exactly the right mixture.

While FASTUS is an elegant engineering achievement, the whole host of linguistic problems that were bypassed are still out there, and will have to be addressed eventually for more complex tasks, and to achieve higher performance on simple tasks. It is in the nature of competitive evaluations is that they force everyone to deal with the easiest problems first. However, the hard problems cannot be ignored forever, and scientific progress requires that they be addressed.

Acknowledgments

This research was funded by the Defense Advanced Research Projects Agency under Office of Naval Research contracts N00014-90-C-0220, and by an internal research and development grant from SRI International.

References

[1] Black, Alan W., 1989. "Finite State Machines from Feature Grammars," in Tomita, ed., *International Workshop on Parsing Technologies*, pp. 277–285.

[2] Church, Ken W., 1980. *On Memory Limitations in Natural Language Processing*, MIT Laboratory of Computer Science Technichal Report MIT/LCS/TR-245.

[3] Hobbs, Jerry R., Douglas E. Appelt, John Bear, Mabry Tyson, and David Magerman, 1992a. "Robust Processing of Real-World Natural-Language Texts", in *Text-Based Intelligent Systems: Current Research and Practice in Information Extraction and Retrieval*, P. Jacobs, editor, Lawrence Erlbaum Associates, Hillsdale, New Jersey, pp. 13-33.

[4] Hobbs, Jerry R., Mark Stickel, Douglas Appelt, and Paul Martin, 1992b. "Interpretation as Abduction", to appear in *Artificial Intelligence*. Also

published as SRI International Artificial Intelligence Center Technical Note 499, December 1990.

[5] Lehnert, Wendy, Claire Cardie, David Fisher, Ellen Riloff, and Robert Williams, 1991. "Description of the CIRCUS System as Used for MUC-3", *Proceedings*, Third Message Understanding Conference (MUC-3), San Diego, California, pp. 223-233.

[6] Pereira, Fernando, 1990. "Finite-State Approximations of Grammars", *Proceedings*, DARPA Speech and Natural Language Workshop, Hidden Valley, Pennsylvania, pp. 20-25.

[7] Pereira, Fernando, and R. Wright, 1991. "Finite-State Approximation of Phrase Structure Grammars", *Proceedings*, 29th Meeting of the Association for Computational Linguistics, Berkeley, California, pp. 246–255.

[8] Sundheim, Beth, ed., 1991. *Proceedings*, Third Message Understanding Conference (MUC-3), San Diego, California, May 1991. Distributed by Morgan Kaufmann Publishers, Inc., San Mateo, California.

[9] Sundheim, Beth, ed., 1992. *Proceedings*, Fourth Message Understanding Conference (MUC-4), McLean, Virginia, June 1992. Distributed by Morgan Kaufmann Publishers, Inc., San Mateo, California.