FASTUS: A System for Extracting Information from Text*

**Jerry R. Hobbs, Douglas Appelt, John Bear,
David Israel, Megumi Kameyama, and Mabry Tyson**

SRI International
333 Ravenswood Avenue
Menlo Park, CA 94025

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 (Japanese is under development), for entry into a database, and potentially for other applications. It works essentially as a set of cascaded, nondeterministic finite state automata.

FASTUS is most appropriate for *information extraction* tasks, rather than full text understanding. That is, it is most effective for text-scanning tasks where

- Only a fraction of the text is relevant.
- There is a pre-defined, relatively simple, rigid target representation that the information is mapped into.
- The subtle nuances of meaning and the writer's goals in writing the text are of no interest.

THE STRUCTURE OF THE MUC-4 FASTUS SYSTEM

The operation of FASTUS is comprised of four steps.

1. Triggering: Sentences are scanned for key words to determine whether they should be processed further.
2. Recognizing Phrases: Sentences are segmented into noun groups, verb groups, and particles.
3. Recognizing Patterns: The sequence of phrases produced in Step 2 is scanned for patterns of interest, and when they are found, corresponding "incident structures" are built.
4. Merging Incidents: Incident structures from different parts of the text are merged if they provide information about the same incident.

*This research was supported in part by the Defense Advanced Research Projects Agency under Contract ONR N00014-90-C-0220 with the Office of Naval Research, in part by NTT Data, and in part by an SRI internal research and development grant. The views and conclusions contained in this document are those of the authors and should not be interpreted as necessarily representing the official policies, either expressed or implied, of the Defense Advanced Research Projects Agency of the U.S. Government.

Many systems have been built to do pattern matching on strings of words. One crucial innovation in the FASTUS system has been separating that process into the two steps of recognizing phrases and recognizing patterns. Phrases can be recognized reliably with purely syntactic information, and they provide precisely the elements that are required for stating the patterns of interest.

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

AN EXAMPLE

The task in the MUC-3 and MUC-4 (Message Understanding Conference) evaluations of text processing systems was to scan news reports and extract information about terrorist incidents, in particular, who did what to whom. The following sentence occurred in one report:

> 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.

1. Triggering: This sentence is triggered because it has a number of key words, including "terrorist", "killing", and "FMLN".

2. Recognizing Phrases: Step 2 segments the sentence into the following phrases:

Noun Group:	Salvadoran President-elect
Name:	Alfredo Cristiani
Verb Group:	condemned
Noun Group:	the terrorist 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 phrases that are recognized are names, the noun group, or the noun phrase up through the head noun, the verb group, or the verb together with its auxilliaries and any trapped adverbs, and various particles, including prepositions, conjunctions, relative pronouns, the word "ago", and the word "that" which is treated specially because of the ambiguities it gives rise to. Essentially the full complexity of English noun groups and verb groups is accommodated.

This phase of the processing gives very reliable results—better than 96% accuracy on the data we have examined.

3. Recognizing Patterns: In the example, two patterns are recognized in the sequence of phrases:

<Perpetrator> <Killing> of <HumanTarget>

and

<GovtOfficial> accused <PerpOrg> of <Incident>

Two corresponding incident structures are constructed:

Incident:	KILLING
Perpetrator:	"terrorist"
Confidence:	–
Human Target:	"Roberto Garcia Alvarado"

and

Incident:	INCIDENT
Perpetrator:	FMLN
Confidence:	Suspected or Accused by Authorities
Human Target:	–

Altogether for the MUC-4 application, about one hundred patterns were recognized.

4. Merging Incidents: These two incident structures are merged into a single incident structure, containing the most specific information from each.

Incident:	KILLING
Perpetrator:	FMLN
Confidence:	Suspected or Accused by Authorities
Human Target:	"Roberto Garcia Alvarado"

In the MUC-4 system, 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.

SKIPPING COMPLEMENTS

Pattern-matching approaches have often been tried in the past, without much success. We believe that our success was due to two key ideas. The first, as stated above, is the use of *cascaded* finite-state automata, dividing the task at the noun group and verb group level. The second is our approach to skipping over complements.

One significant problem in pattern-matching approaches is linking up arguments with their predicates when they are distant in the sentence, for example, linking up the subject noun group with the main verb when the subject has a number of nominal complements. One technique that has been tried is to skip over up to some number of words, say, five, in looking for the subject's verb. One trouble with this is that there are often more than five words in the subject's nominal complement. Another trouble is that in a sentence like

The police reported that terrorists bombed the Parliament today.

this technique would find "the police" as the subject of "bombed".

Our approach is to implement knowledge of the grammar of nominal complements directly into the finite-state pattern recognizer. 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. This is how, in the above example, we were able to recognize Cristiani as the one who was accusing the FMLN of the crime.

THE PERFORMANCE OF FASTUS

On the MUC-4 evaluation in June 1992, FASTUS was among to top few systems, even though it had only been under development for five months. On the TST3 set of one hundred messages, FASTUS achieved a recall of 44% and a precision of 55%. The full results of the MUC-4 evaluation can be found in Sundheim (1992).
Moreover, FASTUS is an order of magnitude faster than any other comparable system. In the MUC-4 evaluation it was able to process the entire test set of 100 messages, ranging from a third of a page to two pages in length, in 11.8 minutes of CPU time on a Sun SPARC-2 processor. The elapsed real time was 15.9 minutes. In more concrete terms, 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.
This fast run time translates directly into fast development time. FASTUS became operational on May 6, 1992, and we did a run on a set of messages that we had not trained on, obtaining a score of 8% recall and 42% precision. At that point we began to train the system on 1300 development texts, adding patterns and doing periodic runs on the fair test to monitor our progress. This effort culminated three and a half weeks later on June 1 in a score of 44% recall and 57% precision. (Recall is percent of the possible answers the system got correct; precision is percent of the system's answers that were correct.) Thus, in less than a month, recall went up 36 points and precision 15 points.
A more complete description of FASTUS and its performance is given in Hobbs et al. (1992).

RECENT EXTENSIONS

We are currently extending the FASTUS system in three ways:

- We are developing a convenient interface that will allow users to define patterns more easily.
- We are implementing a Japanese language version of FASTUS.
- We are applying the system to a new domain -- extracting information about joint ventures from news articles.

The last of these will be the subject of our MUC-5 paper. The other two are described here.

THE INTERFACE

The original version of FASTUS has been augmented with a convenient graphical user interface for implementing or extending an application, employing SRI's Grasper system (Karp et al., 1993). We expect this to speed up development time for a new application by a factor of three or four. Moreover, whereas before now only a system developer could implement a new application, now virtually anyone should be able to.
In a specification interface for FASTUS, there needs to be convenient means for performing four tasks:

1. Defining target structures.
2. Defining word classes.
3. Defining state transitions.
4. Defining merge conditions.

We have done nothing yet in the first two areas, since everyone currently working with the system is fluent in Lisp. Target structures are defined with defstruct, word classes with defvar. As we acquire users who are not programmers, it will be straightforward to implement convenient means for these tasks.
The Grasper-based graphical interface provides a convenient means for creating, examining, editing, and destroying nodes and links in the graphs representing the finite-state automata. Each link is labelled with the tokens that cause that transition to take place. Nodes have associated with them sequences of instructions that are executed when that node is reached. These instructions typically fill slots in the target structures, and they can be conditionalized on what link the node was reached from, allowing greater economy in the finite-state machines.
In addition, the interface allows the graphs at each level to be modularized in whatever fashion the user desires, so that at any given time, the user can focus on only a small portion of the total graph. There are also convenient means for saving and compiling the graphs after changes have been made.
Perhaps the hardest problem in the information extraction task is defining when two target structures can be merged. This is, after all, the coreference problem in discourse, well-known to be "AI-complete". We have developed a kind of algebra on the target structures. The user

can define abstract data types, including number ranges, date ranges, locations, and strings. Comparison operations can then be defined for each of these data types, returning values of Equal, Subsumes, Inconsistent, and Incomparable. Combination operations can also be defined. For example, the combination of two number or date ranges is the more restrictive range. For strings, the combination depends on the semantic categories of the heads of the strings. If one is more specific than the other, the more specific term is the result of the combination. There are three types of actions that be performed after doing a comparison. The items can be merged or combined. If they are incomparable and if the slot in the target structure admits compound entries, the two can simply be added together. Or the unification of the two items can be rejected.

This algebra of target structures gives us a very clean treatment of what in the MUC-4 system was often very ad hoc.

FASTUS has been restructured somewhat as well since MUC-4. A Tokenizer Phase has been added. Its input consists of ascii characters and it output is tokens, usually words, numerals, and punctuation marks. This phase gives the user control over the lowest level of input, so that special rules can be encoded for abbreviations, numbers with radix other than 10, and other such phenomena. The most common tokenizations are, of course, already implemented.

A Preprocessor Phase has also been added. This incorporates the multiword handling that was done in the Phrase Recognition phase of the first version of FASTUS. It also allows the user to customize automata for dealing, for example, with names that have a different given-name family-name order and with names of nonhuman entities that have internal structure significant to the domain, such as company names.

The treatment of appositives, conjunctions, and "of" prepositional phrases was originally done in the Pattern Recognition phase. This has now been separated out into a Combining Phase for a treatment that is more perspicuous and hence more convenient for the user.

JAPANESE FASTUS

We are also developing a Japanese version of FASTUS. The initial application is for extracting a summary of spoken dialogues, input in Roman characters, in the domain of conference room reservations. Summarizing goal-oriented dialogues can be achieved by filling a predefined summary template, and any digressions in the dialogue content can be ignored. Summarization is then an example of expectation-driven information extraction performed by FASTUS.

Despite the dissimilarity between the English and Japanese languages, the basic FASTUS architecture consisting of four phases can be applied to the processing of Japanese. The phrase recognition phase (phase II) recognizes noun groups, verb groups, and particles. The phrase combination phase (phase III) recognizes the "NG no NG" phrases (similar to the English "of" phrases) and NG conjunctions that are of interest to the given domain. The incident recognition phase (phase IV) recognizes those utterance patterns that contain key information relevant to the summary template. Because the input is spontaneous dialogues rather than written news reports, we will have a dialogue managing module after the incident recognition phase in order to combine information contained in successive dialogue turns—for instance, question-answer pairs and request-confirmation pairs. We have implemented phases II and III, and phase IV will be in place shortly.

The main complexity of summarization in this room reservation domain is in the use of temporal expressions and in the dynamics of negotiation between the two speakers. Written news reports typically report past events whose resulting states are already known. Spoken dialogues, however, progress through a sequence of negotiations where the speakers express their desires, possibilities, impossibilities, concessions, acceptances, and so forth. This is a considerable challenge to the structure merging routine of FASTUS.

For the MUC-5 participation, the Japanese FASTUS system will be extended for the new domain of joint ventures and the new input type of written news reports in Japanese characters.

SUMMARY

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. It was among the top few systems in the MUC-4 evaluation.
- 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.

We believe that the FASTUS technology can achieve a level of 60% recall and 60% precision on information extraction tasks like that of MUC-4. Human coders do not

agree on this task more than 80% of the time. Hence, a system working ten times as fast as humans do can achieve 75% of human performance. We believe that combining this system with a good user interface could increase the productivity of analysts by a factor of five or ten in this task.

This of course raises the question about the final 25%. How can we achieve that? We believe this will not be achieved until we make substantial progress on the long-term problem of full text understanding. This cannot happen until there is a long-term commitment that makes resources available for innovative research on the problem, research that will almost surely not produce striking results on large bodies of text in the near future.

Absent such an environment, our immediate plans are to spend about two months bringing our MUC-5 system to and beyond the level of our MUC-4 system, and then to explore the important research question of how much of full text understanding can be approximated by the finite-state approach. The following observations are very suggestive in this regard.

We believe that the most promising approach for full text understanding is the "Interpretation as Abduction" approach elaborated in Hobbs et al. (1993). There are three basic operations in this approach, and each of them can be approximated in FASTUS technology. First, the syntactic structure is recognized and a logical form is produced. The corresponding operation in FASTUS is the recognition of phrases, that part of syntax that can be done reliably. Second, the logical form is proven abductively by back-chaining on axioms of the form

$$(\forall a, b) Y(a, b) \supset X(a, b)$$

This can be approximated by adding further patterns: In addition to having a pattern for

A X'ed B

we would also have a pattern for

A Y'ed B

Third, redundancies are spotted and merged to solve the coreference problem. As pointed out above, this is approximated in FASTUS by the operation of merging incidents.

However, it must be realized that much of the success of the FASTUS approach is in the clever ways it ignores much of the irrelevant information in the text. As we deal with texts in which more and more of the information is relevant, this advantage could well be lost, and a genuine, full text-understanding system will be required.

REFERENCES

1. Hobbs, Jerry R., Douglas E. Appelt, John Bear, David Israel, and Mabry Tyson, 1992. "FASTUS: A System for Extracting Information from Natural-Language Text", SRI Technical Note 519, SRI International, Menlo Park, California, November 1992.
2. Hobbs, Jerry R., Mark Stickel, Douglas Appelt, and Paul Martin, 1993. "Interpretation as Abduction", to appear in *Artificial Intelligence Journal*. Also published as SRI Technical Note 499, SRI International, Menlo Park, California. December 1990.
3. Karp, Peter D., John D. Lowrance, Thomas M. Strat, David E. Wilkins, 1993. "The Grasper-CL Graph Management System", Technical Note No. 521, Artificial Intelligence Center, SRI International, January 1993.
4. Sundheim, Beth, ed., 1992. *Proceedings*, Fourth Message Understanding Conference (MUC-4), McLean, Virginia, June 1992. Distributed by Morgan Kaufmann Publishers, Inc., San Mateo, California.