The SRI MUC-5 JV-FASTUS Information Extraction System

Douglas E. Appelt, Jerry R. Hobbs, John Bear,
David Israel, Megumi Kamayama, Mabry Tyson

February 24, 2004

1 Introduction and Background

SRI International developed an information extraction system called `FASTUS`, a permuted acronym standing for "**F**inite **S**tate **A**utomata-based **T**ext **U**nderstanding **S**system for application to general information extraction tasks. The choice of acronym is, however, unfortunately somewhat misleading, because `FASTUS` is an *information extraction* system, not a text understanding system. The former problem is a much simpler, more tractable problem that is characterised by a relatively straightforward specification of information to be extracted from the text that changes slowly over time, if at all, with only a fraction of the text being relevant to the extraction task, and with the author's underlying goals and nuances of meaning of little interest. In contrast, a text understanding task is to recover all of the information that there is in a text, including that which is only implicit in what is actually written. All the richness of natural language becomes fair game, including metaphor, metonymy, discourse structure, and the recognition of the author's underlying intentions, and the full interplay between language and world knowledge becomes central to the task.

Text understanding is extremely difficult, and presents a number of research problems that have not yet been adequately solved. On the other hand, the relative simplicity of the information extraction task means that the full complexity of natural language need not be confronted head-on. In fact, much simpler mechanisms can be successfully employed to solve the more constrained problem, and do so in a computationally efficient and conceptually elegant way. It was this insight that led to the development of the FASTUS system that was applied to the task of extracting information from articles about terrorism in Latin America for the MUC-4 evaluation [Hobbs et al., 1992; Appelt et al., 1993].

In contrast to NL-processing systems designed for text *understanding* applications, `FASTUS` does not do a complete syntactic and semantic analysis of each sentence. Instead, sentences are processed by a sequence of nondeterministic finite-state transducers. The output of each level of transducers becomes the input to the next level. Each level of processing produces some new linguistic structure, and perhaps discards some information that is irrelevant to the information extraction task. The non-determinism of the transducers makes it possible to produce local analyses of fragments of the input that can be combined into a complete analysis without the necessity of determining the complete structure of each sentence, when the effort of producing such a structure has little payoff for the task at hand. The non-determinism can also be exploited to produce competing analyses of portions of the text that can be compared, so that the best analysis can be selected for processing at subsequent levels, reducing the combinatoric complexity of the subsequent levels.

When the transducer for the final level enters a final state, the result is a "raw template" that is unified with other raw templates from the current and previous sentences. Finally a post-processor transforms the raw templates into the form required by the specifications of the task.

The basic architecture of the MUC-5 system has evolved in only relatively minor ways from the MUC-4 system. The primary difference between the MUC-4 `FASTUS` system and `JV-FASTUS` is the addition of a user-interface to facilitate the rapid development of the system in a new domain. When we developed the MUC-4 `FASTUS` system, we had extensive experience working in the terrorist domain, since we had adapted the `TACITUS` system to work in that domain for MUC-3. Before this year the questions were open whether the `FASTUS` system provided the basic tools necessary to develop a new information extraction system from scratch in a very short period of time, and whether the `FASTUS` approach would be successful with languages significantly different from English. We believe that our MUC-5 experience enables us to answer both of these questions with a confident "yes."

2 Results of the Evaluation

The `JV-FASTUS` system achieved an error rate per slot fill of 0.70 and richness-normalized error of 0.82 on English joint ventures. The error rate was the third best result reported, and was exceeded only by systems that had significantly longer domain-specific development time than `FASTUS`. The richness-normalized error was the second best of all systems reported. This error rate corresponds to recall of 34% and precision of 56% and an equally-weighted F-metric of 42.67.

The `JV-FASTUS` system also achieved an error rate per slot fill of 0.70 and richness-normalized error of 0.79 on Japanese joint ventures. This was also the third-best error rate of all sights reporting, and was exceeded only by systems that had significantly longer development time. This error rate corresponds to recall of 34% and precision of 62% and an equally-weighted F-meteric of 44.21.

After the MUC-4 conference, SRI embarked on an effort to rationalize our MUC-4 system, refine its overall architecture, and to add a user interface to facilitate the definition and maintenance of the finite-state transducers comprising the system. With the exception of a very skeletal system for English joint ventures based on a corpus of about 10 articles extracted by hand from current issues of the Wall Street Journal, which was used as the basis of a demonstration at the ARPA HLT meeting, no domain-specific development was undertaken until the beginning of April. The first end-to-end test of the English joint ventures system on 100 texts was conducted on April 30, 1993 with the result of an unimpressive error rate of 0.93 (F-measure 6.02)

At the same time, we ran the first end-to-end test of the Japanese version of `JV-FASTUS`. Although we had developed a version of `FASTUS` called `MIMI` for information extraction from Japanese spoken dialogues [ref?], and thus had some experience in processing Japanese with FASTUS, the actual base system for `MIMI` is really the same as the English system, since it operates on a Romaji encoding of speech, rather than on Kanji characters. [Megumi to put some results in here].

We succeeded in raising the system performance from this baseline to our reported results in 3 months of work. We feel that this experience confirms the adequacy of the tools provided by `JV-FASTUS` for the rapid development of information extraction systems in new domains. We stopped development of the system at approximately 3:00 PM on August 1. At that time, our improvement curve was still extremely steep. Work during of the morning of August 1 resulted in a 0.5 point improvement in F-measure. We had not even attempted to produce revenue objects, and our treatment of times and facilities was extremely sketchy. We feel that another week or two of development would have led to significantly improved results.

3 System Architecture

The basic architecture of the English JV-FASTUS system is illustrated in Figure 1. The text is input to the cascade of transducers as a stream of ASCII characters. This ammounts to a decision to treat all text as unformatted, which for the English joint ventures texts is not unreasonable, since these texts contain very little relevant formatted data such as tables, and when they do occur, their format is ideosyncratic. The first transducer is the `TOKENIZER`, which produces symbolic and numeric tokens as output. These symbolic tokens are given to the `PREPROCESSOR`, which recognizes multiword lexical items, and some company and personal names, and produces lexical-items as output. The `PHRASE PARSER` then breaks the input stream into Noun Groups (the part of the noun phrase consisting of determiner, prenominal modifiers and head noun) Verb Groups (auxilliaries, intervening adverbs, with main verb) and particles (single lexical items including conjunctions, prepositions, subordinating conjunctions, and relative pronouns). The `PHRASE` parser also identifies the head of each constituent, which with some minor exceptions, is the only component of the constituent that influences subsequent processing. The `PHRASE COMBINER` takes the phrases output by the `PHRASE PARSER` and combines them into larger phrases of the same type. For example, ad-

Figure 1: The FASTUS Flow of Control

jacent noun groups may be merged into appositives, certain prepositional phrases are attached to their noun groups, and conjunctions of both verb groups and noun groups are combined. The combined phrases are input to the `DOMAIN PATTERN RECOGIZER`, which nondeterministically matches the sentence against patterns that are relevant to the information to be extracted. The by-product of the match is partially instantiated raw templates that are merged by the `MERGER`. Finally a `POST PROCESSOR` puts the raw templates into final form for printing.

3.1 The Walkthrough Example

The text of the system-walkthrough is given in an appendix to this volume. We were dismayed to see that our system did not produce a template in response to the walkthrough text. Closer inspection, however, revealed that the system *had* in fact produced a reasonable analysis for this text, but the analysis was discarded by the `POST PROCESSOR` because it failed a basic consistency check: the joint venture company had to be distinct from all of its parent entities. Experience has shown that a failure to satisfy this condition usually arises due to a failure in the merging process, and it turns out that the score is usually improved by discarding what is likely to be a spurious template. Unfortunately in this case the strategy resulted in discarding basically correct information. We turned off the filter, and reran the example, producing the output listed in Appendix 1.

3.2 The TOKENIZER

The `TOKENIZER` is a simple transducer that accepts ascii characters as input and produces a stream of tokens as output. The tokenizer performs the following functions:

- Groups characters into "words."
- Computes value of numeric tokens.
- Detects abbreviations and determines sentence boundaries.
- Normalizes corporate prefixes and suffixes such as P.T. and Inc.

In case of ambiguity, the ambiguity is resolved in favor of the longest token that can be formed starting at the current position in the input stream.

The walkthrough text does not present any unusual difficulties for the `TOKENIZER`.

3.3 The PREPROCESSOR

The `PREPROCESSOR` accepts the tokens produced by the `TOKENIZER` as input and produces lexical items as output. A lexical item is defined as a token or sequence of tokens that has an entry in the system's lexicon. During this phase, multiwords are recognized. Proper names of individuals, locations and corporations are considered lexical items, and the `PREPROCESSOR` makes the first attempt to recognize them.

Case is very important for disambiguating proper and common nouns in English. In texts with both upper and lower case characters, capitalization provides very useful information about which words can or cannot be parts of names, which is not available in upper-case-only texts. Therefore, the `PREPROCESSOR` uses separate transducers for recognizing personal and corporate names for mixed case and upper-case-only texts.

There are three basic transducers for corporate names. There is a transducer that operates on both upper-only and mixed case texts that recognizes company names that do not appear with a standard suffix like "Inc," or "GmbH." There is a recognizer for mixed case text corporate names that basically accepts all capitalized words preceding a suffix like "Inc," with some heuristics to avoid including capitalized words at the beginning of a sentence that are not part of the name. Upper case only texts present more of a problem, because the simple expedient of accepting any noun group preceding the corporate suffix leads to overgeneration of company names, particularly in cases of lexical ambiguity of the words involved. For example, a sentence like "`ALBION IRON & METAL SAW AN INCREASE IN PROFITS THIS YEAR`" would probably result in "`ALBION IRON & METAL SAW`" as the name of the company, because "saw" can be a noun as well as a verb. To prevent this kind of overgeneration of company names, we restrict the words that can combine to form company names to be a member of a list of product words that are likely to occur in names. "Iron" and "metal" occur on this list, while "saw" does not.

This heuristic for recognizing company names in upper-case-only texts caused the most serious problem we encountered in the walkthrough example. The first sentence of this example is

```
BRIDGESTONE SPORTS CO. SAID FRIDAY IT HAS SET UP A  JOINT VENTURE  IN
TAIWAN  WITH A LOCAL CONCERN AND A  JAPANESE  TRADING HOUSE TO PRODUCE GOLF
CLUBS TO BE SHIPPED TO  JAPAN.
```

It turns out that "`BRIDGESTONE`" is known in the lexicon to be the name of a company, however "`SPORTS`" was not on the list of product words. Therefore, the system recognized "`BRIDGESTONE`" as a company name and as the subject of the sentence, and ignored "`SPORTS CO.`" as an apositive.

When a company name is recognized, it is entered into the lexicon for the duration of the text, together with any possible aliases that can be predetermined. The lexicon is restored to its initial state at the end of a text so any mistakes or perverse company names will have no effect on subsequent processing. For example, if an article mentions "Next, Inc." it is important to recognize "Next" as a company name for the duration of the text, but that could obviously cause havoc with other texts.

In summary, the preprocessor performs the following functions:

- Groups of words comprising multiword lexical items are collected together.
- Company names that are in the system's lexicon, or composed from lexical entries by systematic rules are identified.
- Names of people are identified and grouped with their titiles.
- Groups of words that could possibly be companies, but for which it is not certain that they are companies are flagged as possible company names.

In case of ambiguity, the longest phrase beginning at the current point in the input string is selected.

3.4 The PHRASE PARSER

The next phase accepts the lexical items combined by the preprocessor as input and produces a sequence of phrases as output. The head of each phrase is identified, and if the head of the phrase corresponds to an object in the domain for which a template object is defined, then an object of the appropriate type is associated with the phrase. For example, if the noun group is "the Japanese company," the noun group is associated with an ENTITY object whose NATIONALITY slot is Japan.

The phrase parser constructs phrases that can be reliably described as a regular language. Attachment ambiguities are preserved for later phases where they will either be ignored as irrelevant, or combined on the basis of domain specific patterns when the combination can be done reliably.

The basic grammar of English used in this phase is a superset of that used in the MUC-4 `FASTUS` system. The main differences involve more detailed processing of numbers consisting of mixed numeric and sybolic parts (e.g. 3 million), currency phrases (e.g. DM 2500), and the recognition of bank names. Possible companies are treated as proper nouns and can be combined to form noun groups just as other proper nouns referring to locations, companies or people.

Lexical ambiguity can lead to multiple analyses at the end of the parsing phase. In general, longer phrases are prefered to shorter ones. In mixed case texts, nominals with proper noun heads are preferred to other analyses if they are capitalized. In upper-case-only texts, company names are preferred to other analyses because of the central role that companies play in the joint venture domain. However, in upper-case-only texts, common nouns and verbs are preferred to location names when ambiguity arises, because of the relatively large number of locations in the gazetteer that overlap with ordinary English words.

As an example, here is how the `PHRASE PARSER` ananlyzes the first sentence of the walkthrough example:

```
CN: "BRIDGESTONE " (0,1) Head: BRIDGESTONE
NG: "SPORTS " (1,2) Head: SPORTS
ACTIVE/PASSIVE: "SAID " (3,4) Head: SAID
NG: "FRIDAY " (4,5) Head: FRIDAY
NG: "IT " (5,6) Head: IT
ACTIVE: "HAS SET " (6,8) Head: SET
PREP: "UP " (8,9) Head: UP
NG: "JOINT-VENTURE " (9,12) Head: JOINT-VENTURE
PREP: "IN " (12,13) Head: IN
LOC: "TAIWAN " (13,14) Head: TAIWAN
PREP: "WITH " (14,15) Head: WITH
NG: "LOCAL CONCERN " (15,18) Head: CONCERN
CONJ: "AND " (18,19) Head: AND
NG: "JAPANESE TRADING HOUSE " (19,23) Head: HOUSE
INF: "TO PRODUCE " (23,25) Head: PRODUCE
NG: "GOLF CLUBS " (25,27) Head: CLUBS
INF: "TO BE " (27,29) Head: BE
ACTIVE/PASSIVE: "SHIPPED " (29,30) Head: SHIPPED
PREP: "TO " (30,31) Head: TO
LOC: "JAPAN " (31,32) Head: JAPAN
```

At this point the system now has entity objects representing a company named "BRIDGESTONE," a "JOINT-VENTURE" and a "LOCAL CONCERN." The local concern has a location of "TAIWAN" because that was the most recently mentioned location. The system did not realize that a noun group with the head "HOUSE" could refer to a company, so no entity is created for "JAPANESE TRADING HOUSE."

3.5 The PHRASE COMBINER

The PHRASE COMBINER attempts to simplify the job of the final domain pattern recognizer by combining phrases from the initial parse into larger phrases whenever this is feasible. This combination takes place in a hierarchy of stages, so that various combination operations can be prioritized. For example, the attachment of certain prepositional phrases is performed before conjunction combination, so conjunction can apply to noun groups with prepositional phrases attached.

Each level of the phrase combination phase has two subphases: a defeat subphase, and a pattern matching subphase. If a pattern in the defeat subphase matches the input, then that string is prevented from matching any pattern in the matching subphase. For example, in general, "for" and "of" prepositions attach almost always to their closest noun group. These attachments are routinely made *except* in two cases: (1) there is a verb that explicitly subcategorizes for a "for" or "of" complement, or (2) the subject and object of the preposition form an important domain pattern that is recognized during the next phase (e.g. "the production of golf clubs"). In these cases, defeat patterns are written to match the input and prevent the pp attachment rule from operating.

The `PHRASE COMBINER` performs the following tasks:

- Adjacent location NGs are merged when the result of the merger is consistent with the information in the gazetteer (e.g. "Palo Alto, California").

- Noun groups with company words as heads are combined with appositives, genitives, and prepositions to provide further information about the entity (e.g. "Foobarco, the California company," "Japan's Kobe Steel," or "Aerospatiale of France.") If any of the company names in the input are only possible companies (like Foobarco), matching one of these patterns will cause the possible company to recognized as a company for the duration of the text.

- Appositives and prepositions that associate people with titles and companies are combined, and their semantics processed. (e.g. "John Smith, president and CEO of Foobarco") Any possible company that matches the pattern is promoted to an actual company.

- Conjunctions of company names are combined (e.g. IBM, General Motors, and Foobarco) Again, possible companies can be promoted if they match the pattern.

- Certain patterns that can reliably be used to promote possible companies to actual companies are recognized, even though they don't directly contribute any information to template slots (e.g. "the board of directors of Foobarco")

- Conjoined verb groups are recognized, as well as certain phrases that can be treated by subsequent analysis as complex verb groups (e.g. "manufacture and market", "planning to set up", "announced a plan to form").

- Finally, "of" and "for" prepositions are attached to their adjacent noun groups, and conjoined noun groups are combined, unless defeated by defeat patterns.

As an example of the operation of the PHRASE COMBINER, consider the system's processing of the second sentence of the walkthrough text:

```
CN: "JOINT-VENTURE BRIDGESTONE " (0,5) Head: BRIDGESTONE
NG: "SPORTS " (5,6) Head: SPORTS
LOC: "TAIWAN " (6,7) Head: TAIWAN
```

```
ACTIVE/PASSIVE: "CAPITALIZED " (9,10) Head: CAPITALIZED
PREP: "AT " (10,11) Head: AT
NG: "20 MILLION NEW TAIWAN DOLLARS " (11,16) Head: DOLLARS
ACTIVE: "WILL START " (17,19) Head: START
NG: "PRODUCTION " (19,20) Head: PRODUCTION
PREP: "IN " (20,21) Head: IN
NG: "JANUARY 1990 " (21,23) Head: -DATE-
PREP: "WITH " (23,24) Head: WITH
NG: "PRODUCTION " (24,25) Head: PRODUCTION
PREP: "OF " (25,26) Head: OF
NG: "20000 IRON AND METAL WOOD CLUBS " (26,32) Head: CLUBS
NG: "MONTH " (32,34) Head: MONTH
```

The PHRASE COMBINER combined two noun groups in this sentence. The combination producing "20000 IRON AND METAL WOOD CLUBS" was correct. Unfortunately, because of the problem cited above with the word "SPORTS," the system did not correctly recognize the joint venture company name, and the combiner formed the appositive "JOINT VENTURE BRIDGESTONE" and assigned BRIDGESTONE the role as the joint venture company. Of course, BRIDGESTONE was already identified as one of the parent entities, so this was the source of the mistake that led to discarding the entire analysis of this text. Also, because the previous sentence said that the joint venture was "in Taiwan" we identified Taiwan as the location of Bridgestone.

3.6 The DOMAIN PATTERN RECOGNIZER

The DOMAIN PATTERN RECOGNIZER does the most critical work of the system by recognizing phrases that establish the most important relationships to be extracted. The DOMAIN PATTERN RECOGNIZER takes the output of the PHRASE COMBINER as input, and produces raw templates as output.

The PATTERN RECOGNIZER of the MUC-4 system had only one subphase, but it was recognized that because of the limited development time available we could not possibly account for all the possible ways joint venture relationships could be expressed. Therefore it was decided to implement the JV-FASTUS PATTERN RECOGNIZER as a multiphase process. The output of the earlier phases would be kept by the system only as long as they were consistent with output found in the later phases. Thus the earlier phases of the PATTERN RECOGNIZER could be used to implement extremely general, loose patterns that could serve as defaults that could be defeated by the output of more precise, specific patterns at higher levels.

Inspection of the corpus indicated that there are three basic, general patterns that indicate joint venture relationships with surprisingly high reliability. They are

- <company-name>$^{++}$ "joint venture" <company-name>
- <company-name> "joint venture" <company-name>$^{++}$
- <company-name>* "joint venture" <company-name>*

The first pattern means that there are at least two occurrances of company names preceeding the words "joint venture" (ignoring all other words) and a single company name following the word "joint venture." The parent entities are the first set of companies, and the joint venture entitiy is the singular one. Typical instances of this pattern are "The Toyota - General Motors joint venture, NUMMI..." and "IBM and Intel formed a joint venture called Foobarco." The second pattern is the "passive" variant of the first (although verb groups and their properties are completely ignored!) which matches sentences like "Foobarco is a joint venture formed by IBM and Intel." Finally the third pattern matches sentences that don't meet the number constraints of the above pattern. In that case, all of the entities are parents. An example is "IBM formed a joint venture with Intel to produce mainframes in Timbuktu."

It is, of course, easy to think of counterexamples to the above patterns. The above patterns help recall much more than they hurt precision, however, because they are only defaults that can be defeated by more precise information. We were initially skeptical that the inclusion of such vague patterns would actually enhance system performance. However, a test showed that they improved the system's F-metric by approximately 6 points.

We eventually settled on three levels for the PATTERN RECOGNIZER. The first level consisted of the above patterns, the second level consisted of a very general pattern for recognizing ownership percentages with active verbs (which, like the above patterns, never actually examined the verbs or their properties), and the third level included a similar pattern for passive ownership percentages (which is more constrained because of the frequent use of the preposition "by") together with more obviously motivated patterns for joint ventures and products.

In the walkthrough example in the first sentence, the system recognized the pattern "`BRIDGESTONE ... SAID IT HAS SET UP A JOINT VENTURE ... WITH A LOCAL CONCERN`." This pattern led to a tie-up relationship with Bridgestone and a company as parent entities. The adverbial "`IN TAIWAN`" was recognized nondeterministically by a different pattern which caused "`TAIWAN`" to be recorded as a default location for the joint venture, and to provide a referent for "local" in "`LOCAL CONCERN`." As mentioned previously, the system did not realize that "`JAPANESE TRADING HOUSE`" was a company.

3.7 The MERGER

The `MERGER` operates at the end of each sentence in two steps: first all the raw templates that are found in a single sentence are merged to the extent possible, and then the remaining templates are merged with any templates from previous sentences.

There are two types of merge operations: full merges on templates of like types, and default merges on templates of different types. Full merges are like unification operations. The merger merges each slot of the two templates recursively, each time determining the best alignment for elements of a slot when the slots can contain multiple fills.

Default merges involve templates of different types. If it is possible for the template of one type to fill a slot, or merge with the slot contents of one of the slots in the other template, and certain other conditions are satisfied, then the merger is accomplished by filling in the appropriate slot. Default merging allows the combination of information from disparate parts of the text into a single tie-up schema. Default merges are allowed as long as the parts occur reasonably near each other in the text. We have found the best results with allowing default merges over a distance of two sentences.

In the walkthrough example, as previously mentioned, the entity for Bridgestone was merged with the joint venture company because of the appositive, and because the company name was incorrectly recognized. Then, the pattern recognizer recognizes the sequences "`BRIDGESTONE ... CAPITALIZED AT 20 MILLION NEW TAIWAN DOLLARS`" leading to an instantiation of a tie-up relationship with the joint venture company *BRIDGESTONE*, and an OWNERSHIP object giving the capitalization, and "`PRODUCTION OF 2000 IRON AND ''METAL WOOD'' CLUBS`" as an activity and industry with appropriate industry-type and product/service slot fills. The tie-up relationship combines in a full merge with the tie up relationship from the previous sentence, and since nearness constraints are satisfied, the activity object is attached to the tie-up relationship at this time.

The next sentence partially matches the passive-ownership pattern, however full recognition of the pattern was blocked by the failure to correctly recognize the company name "`UNION PRECISION CASTING CO.`" and the erroneous attachment of "`AND THE REMAINDER`" to the previous noun group as a conjunction. The result was a tie up relationship with an ownership template attributing 75% ownership to Bridgestone, which merged in a full merge with the previously found tie-up relationship and ownership. This example illustrates the crucial importance of recognizing company names in this domain. If the company names had been correctly recognized here, the system's output would have been nearly perfect. As a direct result of name recognition failure, compounded errors led to a much less satisfactory result.

Finally, spurious activity and industry templates are produced from the next sentence, which recognizes "`PRODUCTION OF GOLF CLUB PARTS`" and attaches it to the tie-up relationship in a default merge, because nearness constraints are satisfied.

3.8 The POST PROCESSOR

The output of the `PATTERN RECOGNIZER` is raw templates. These templates match the structure of the officially specified templates rather closely, but they contain enough differences to require normalization of the output before printing so they will meet the specifications of the task. This task falls to the `POST PROCESSOR`. The post processor is a rather complicated and task-specific piece of code which performs several, mostly uninteresting functions. The following tasks are assigned to the `POST PROCESSOR`:

- ENTITY-RELATIONSHIP objects are generated for entities involved in joint ventures. (Subordinate ENTITY-RELATIONSHIPs are generated as a result of patterns recognized when processing the text.)
- Ordered pair slots are constructed where required (The system treats ordered-pair fills as full objects, as they were in the original TIPSTER specifications, because this makes the merging algorithm simpler).
- String fills are extracted from the original text, rather than printed in the normalized, upper case form used by JV-FASTUS.
- Company names are extracted from the original text and normalized to ensure compliance with the specifications.
- Locations are disambiguated and normalized using information from the gazetteer.
- SIC codes for product-service strings are generated. Associating these codes with strings is really black magic, and the keys are very inconsistent, and in some cases clearly wrong on these. We fill them in for those cases where we feel we can guess the right answer at least 50 percent of the time.
- Dates are normalized and printed according to specifications.

4 Conclusion

Our experience with the MUC-5 evaluation leaves us believing more strongly than ever that the `FASTUS` system is the best approach to the text information extraction problem, and that the technology is now ripe for application to real-world problems. Our experience has shown that

- Information can be extracted from text very rapidly.
- It is possible to bring a system up to a high level of performance very quickly given the right set of tools. `FASTUS` provides us with the tools we need.

- The system architecture can be straightforward, conceptually simple and easily understood.

Future research in this area should be directed toward the application of this technology to increasing analyst productivity. Open questions are how a generic information extraction system like `FASTUS` can be customized to a particular application like `JV-FASTUS` without extensive intervention from its original developers. Interesting ideas to investigate are how a system can automatically acquire information from a corpus of related texts, or from learning and generalizing from observing analysts annotating texts.

Appendix I

The output produced by `JV-FASTUS` on the walkthrough text, with consistency filtering turned off:

```
<TEMPLATE-0592-1> :=
   DOC NR: 0592
   DOC DATE: 241189
   CONTENT: <TIE_UP_RELATIONSHIP-592-35>
<TIE_UP_RELATIONSHIP-592-35> :=
   TIE-UP STATUS: EXISTING
   ENTITY: <ENTITY-592-7>
           <ENTITY-592-1>
   JOINT VENTURE CO: <ENTITY-592-1>
   OWNERSHIP: <OWNERSHIP-592-13>
   ACTIVITY: <ACTIVITY-592-22>
             <ACTIVITY-592-11>
<ENTITY-592-7> :=
   TYPE: COMPANY
   ENTITY RELATIONSHIP: <ENTITY_RELATIONSHIP-592-2>
<ENTITY-592-1> :=
   NAME: BRIDGESTONE
   LOCATION: Taiwan (COUNTRY)
   TYPE: COMPANY
   ENTITY RELATIONSHIP: <ENTITY_RELATIONSHIP-592-2>
<OWNERSHIP-592-13> :=
   TOTAL-CAPITALIZATION: 20000000 TWD
   OWNED: <ENTITY-592-1>
   OWNERSHIP-%: (<ENTITY-592-1> 75)
```

```
<ACTIVITY-592-22> :=
   INDUSTRY: <INDUSTRY-592-22>
   ACTIVITY-SITE: (Taiwan (COUNTRY) -)
<ACTIVITY-592-11> :=
   INDUSTRY: <INDUSTRY-592-11>
   ACTIVITY-SITE: (Taiwan (COUNTRY) -)
<ENTITY_RELATIONSHIP-592-2> :=
   ENTITY1: <ENTITY-592-7>
            <ENTITY-592-1>
   ENTITY2: <ENTITY-592-1>
   REL OF ENTITY2 TO ENTITY1: CHILD
   STATUS: CURRENT
<INDUSTRY-592-22> :=
   INDUSTRY-TYPE: PRODUCTION
   PRODUCT/SERVICE: (- "GOLF CLUB PARTS")
<INDUSTRY-592-11> :=
   INDUSTRY-TYPE: PRODUCTION
   PRODUCT/SERVICE: (- "20,000 IRON AND "METAL WOOD" CLUBS")
```