TRW:
DESCRIPTION OF THE DEFT SYSTEM AS USED FOR MUC-5

WILLIAM W. NOAH, Ph.D.
ROLLIN V. WEEKS

TRW SYSTEMS DEVELOPMENT DIVISION
ONE SPACE PARK
REDONDO BEACH, CA 90278
R2/2186

BACKGROUND

For the past three years, TRW has been developing a text analysis tool called DEFT-- Data Extraction from Text. Based on the Fast Data Finder (FDF), DEFT processes large volumes of text at very high speeds, identifying patterns which serve as indicators for the presence of relevant objects, relationships, or concepts in the data. These indicators are processed by a series of system-supplied utilities or custom-written functions which refine the data and re-formulate it into frames which can be presented to a user for review, editing, and submission to a downstream application or database.

Superficially, DEFT resembles a Natural Language Understanding (NLU) system; however, there are key differences. DEFT entertains very limited goals in the processing of natural language input. Although DEFT processes unconstrained input, it is looking for textual entities which are tightly constrained and presented to the system as a list of expressions or in a powerful pattern specification language. It exploits expectations about how a small set of entities will be expressed to reduce the amount of computation required to locate those-- and only those-- entities. The broader question of the "meaning" of the text in the document is bypassed in favor of rapid, robust processing that can be readily moved from domain to domain. As long as the input for a particular domain is sufficiently predictable, data extraction with a satisfactory level of recall and precision for many applications can be achieved. We are currently installing three DEFT systems for a United States government agency; initial reviews have been highly favorable.

Our involvement in MUC-5 derives from a request by the government to turn DEFT to a COTS product, with the intent of having a fully-supported version of the system by the end of the year. An analysis of the broader commercial and government market for text extraction suggested that the scope of problems that DEFT should be able to address needed to be expanded; however, it was established that replication of the on-going research and development work in the NLU community was an inappropriate role for our development group. Rather, we wanted DEFT to be able to integrate with systems already developed or in development for functionality which falls outside the narrow boundaries of DEFT's pattern-based capabilities. At the same time, DEFT's ability to express patterns needed to be extended from it's current, highly effective means for defining "atomic" patterns to the definition of patterns in relationship to each other, permitting simple syntactic information to be added to DEFT's lexical knowledge. Thus, DEFT would have the potential to find entities not expressly defined in a lexicon, improve its ability to correctly determine the relation between entities, and decrease the overgeneration that tends to be associated with approaches that rely exclusively on pattern matching.

A mechanism was selected for enhancing pattern specification which was felt to be compatible with the notion of integrating DEFT with third-party systems. As will be described in some detail, DEFT is intrinsically an engineering shell which is intended to facilitate such integration while making its rapid pattern-matching services available to the other system components. Unfortunately, the software implementing this concept was not available at the time of the final MUC-5 evaluation, the results of which therefore serve only to confirm our expectations that the recognition of "simple" (i.e. isolated) patterns is woefully insufficient for complex data extraction problems.

While we regret that the capabilities of the extended version of DEFT could not be demonstrated for MUC-5, we feel that the outcomes justify our belief that real-world message understanding problems necessitate an engineering solution that can pit a choice of technologies against the specific problem at hand-- different technologies being optimum for different tasks. We believe that DEFT's success in handling simple data extraction problems can be extended, and that DEFT is well-suited to a role as an integrator of text analysis capabilities. It is toward this end that we are focusing our on-going productization efforts.

SYSTEM DESCRIPTION

It is convenient to envision DEFT as a pipeline, as shown in Figure 1. At the head is a standardized document interface to message handling systems. At the tail is a process which generates frames and distributes these to the appropriate destinations on the basis of content. In between is a series of text analysis "filters" which apply DEFT lexicons (pattern searches) against the text (using the FDF) and call specific extraction functions to process the textual fragments located by the lexicons. All processes are controlled by means of external configuration files and a "workbench" which contains tools for interacting with DEFT and the data DEFT extracts. We will describe each of these major components in turn.

The Document Interface: Message Queuing. It is assumed that DEFT will be embedded in an existing automated message handling (AMH) system. DEFT's interface with these systems is called Message Queuing (MQ). Text is typically disseminated to MQ (e.g. by a messaging system like TRW's ELCSS or KOALA that receives government cables, wire service input, etc.) on the basis of subject matter, source, structure, or other characteristic with salience for how the message's language will be analyzed. MQ can also accommodate documents loaded from other sources, such as native wire services, an existing full-text database, CD-ROM, OCR, and so on. Text is assumed to be in ASCII or extended ASCII; in the near-future, DEFT will build on work currently underway to allow the FDF to accommodate Unicode for foreign character sets, such as Japanese. Structural features, such as document boundaries, sentence boundaries, paragraphs, tabularization, encoded tags (such as SGML), embedded non-textual media, etc. can be defined for a particular document class using DEFT specification files.

MQ utilizes a configuration file to assign a processing thread tailored to the problem domain to each category of document classified by the dissemination system or by whatever means (including manual) is used to route documents to DEFT. Documents are associated with a processing thread by placing them in a particular MQ "in-basket" (a standard Unix directory). Each in-basket is polled periodically, using a set of criteria (time and number of messages since the last processing thread was initiated) defined in the configuration file.

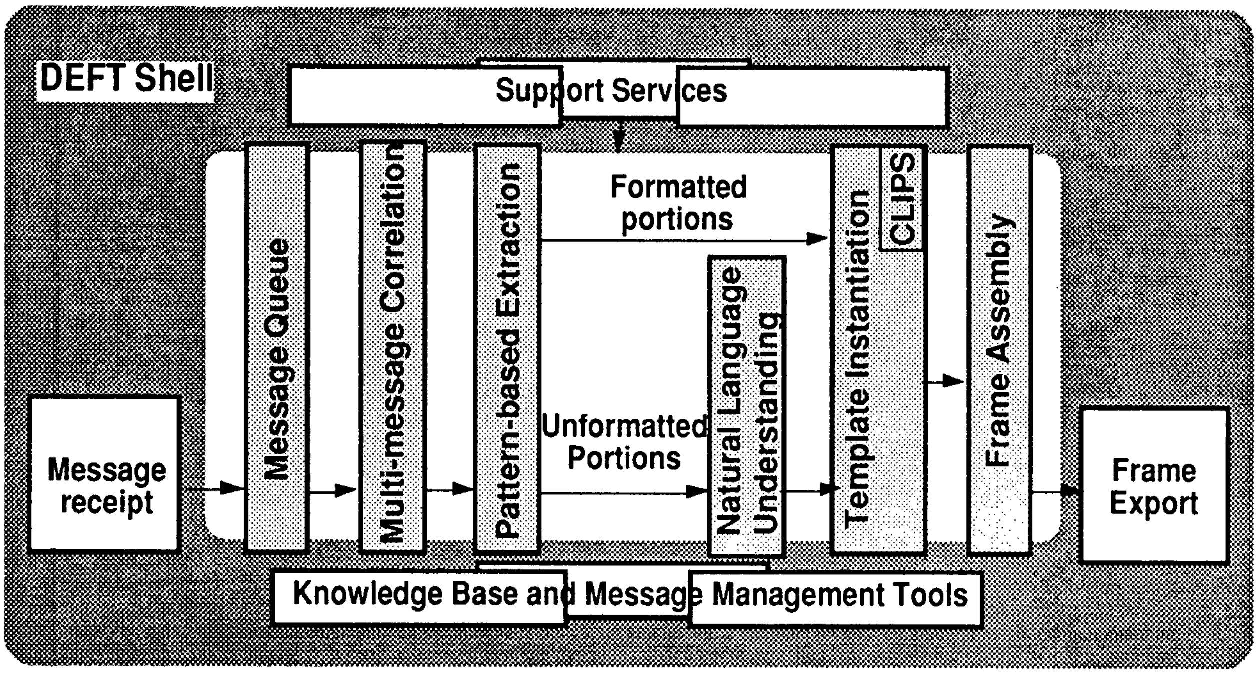

Figure 1: DEFT Functional Architecture

Extracting Data: Text Analysis Filters. When MQ assigns a document to a processing thread, it is subjected to a sequence of procedures which operate on the text to locate patterns of interest and use these patterns as a guide to extract the data required for a particular problem domain. This sequence of processes determines what is extracted and how it is extracted. The sequence is defined as an ordered list of "extraction phases" in a configuration file. This list can be changed at any time to substitute or add new extraction phases to refine a text processing thread. New threads can be modeled on existing ones, facilitating transitions to new problem areas.

Each extraction phase is an executable program. The behavior of a phase is dependent on the order in which it is called (i.e. its relationship to the phases that have been executed before it) and on parameters which are supplied in the configuration file. In this way, a generalized extraction phase can be configured for a specific analytic objective. DEFT has a library of extraction phases that perform the most elementary analytic processes; new phases are be written on a problem-specific basis. DEFT provides an application programming interface (API) in the form of a library of utilities which allows a custom extraction phase to interact with the data structure which is common to all extraction phases, and which is used to communicate between phases. This structure is the DEFT "Tag File."

The Tag File is a cumulative record of the processing performed by each extraction phase. Each phase receives the Tag File from the preceding phase, and passes it to the next. A "tag" represents a textual pattern identified by DEFT in the text or data created by an extraction function.

Much of the power of DEFT comes from the ability to apply a mixture of extraction phases that is optimally suited for a given class of document and extraction problem. For example, one extraction phase might reason about the relative time of occurrence of events located in the text, basing its analysis on the occurrence of various forms of date/time indicators as well as the presence of such modifiers as "last week," or "three years ago." Another phase might construct corporate names on the basis of the occurrence of a known name or the presence of a designator (e.g. "Inc." or "S.A."). Yet another phase might act upon these names to reason about their potential relationship in a joint venture.

Locating Data: DEFT Lexicons. The patterns that DEFT uses to locate data of interest in the text are contained in DEFT's lexicons. Lexicons serve various purposes: to identify potential frames, to determine the "scope" of a frame in the text (i.e. the boundaries to be used to find data to fill the frame slots), to find the contents for a slot in a frame, to determine structural elements (e.g. sentences, paragraphs, header information), and to set the attributes of a text object (e.g. classification level).

Lexicons are of two types: list and pattern. The list lexicon associates a set of synonyms (or spelling variants) with a given object. It is useful when the complete set of strings associated with an object can be specified. The pattern lexicon is used when the textual variations associated with an object cannot be specified. For example, all possible monetary values cannot be conveniently enumerated, but a single pattern describing monetary values in terms of digits, punctuation, and denomination strings can be constructed.

Associated with lexicon entries are attributes, representing the semantics of the problem domain. An attribute is a characteristic of the object represented in the text by its synonym list or pattern. It might be the normalized form of a name or other data about an object which is useful to map into a frame, such as the country associated with a corporate name. In a list lexicon, these attributes are known explicitly when an entry is created; they are not inferred from the text. In a pattern lexicon, however, the attributes cannot be known in advance because it is not known what exact value will hit against the pattern. For this reason, attributes must be extracted for a pattern lexicon. Attribute extraction is handled by a C or C++ program referred to as an "extraction function." For example, given the location of a corporate designator, a function might reconstruct the corporate name.

The success of a data extraction system that relies on pattern matching and string finding depends on how exhaustively it can search for the variations expected in input language. DEFT has proved successful in its current applications in part because its lexicons can be extremely large, thanks to the capabilities (in terms of both functionality and performance) of the FDF.

Searching Text for Lexicon Entries: The FDF. DEFT uses the TRW-developed Fast Data Finder to rapidly locate instances of a potentially enormous set of patterns in the input text. The power of the FDF originates in two ways: the hardware architecture and the expressiveness of its Pattern Specification Language (PSL).

The current generation FDF-3, now a COTS product manufactured by Paracel, Inc., uses a massively parallel architecture to stream text past a search pattern at disk speeds (currently 3.5-million characters/second using a standard SCSI disk).

Searches are compiled into micro code for a proprietary chip set which can accommodate up to 3,600 simultaneous character searches or Boolean operations. Lexicons are broken into "pipelines" which fully fill the chip set; each pipeline is run against all of the text in the set of documents currently being processed. MQ batches messages as they come in so as to optimize the use of the FDF-- larger message sets are processed more efficiently than several smaller ones. The tradeoff between batching and "real-time" processing can be independently balanced in the MQ configuration file for each in-basket and processing thread.

Search patterns are specified in PSL. Because the FDF uses a streaming approach, PSL is not dependent on word boundaries. Extremely complex patterns can be expressed, which can include such features as error tolerance, sliding windows, multiple wildcard options, nested macros, character masking, ranging, and the usual Boolean operations. Features that support "fuzzy matching," like error tolerance, are extremely important for handling "noisy" input.

Output Generation: Frame Assembly and Routing. When the filters that comprise a processing sequence have executed, the Tag File is passed to the "Frame Assembly and Region Routing" (FARR) module. This program, which constitutes the "tail" of the DEFT pipeline, assembles the data elements generated during the analysis thread into frames based on an external definition file. This file specifies which slots are associated with which frames, how to transform a data value for display to the user (e.g. normalize "England" to "United Kingdom"), how to transform a value for storage in a downstream database (e.g. abbreviate "England" as "UK"), how to validate a data value, whether a data type can be multiply-occurring, and so on.

One issue that arises during frame assembly is *when* to associate a data value with an instance of the frame class for which it is defined. In DEFT, this operation is associated with "scoping." Scoping is the process of determining the extent in the text of a concept associated with a pattern. For example, if a pattern of words indicative of a joint venture is found, the scope of the "tie-up" frame might be taken to be the location of the pattern plus or minus two sentences. The unit of scoping (in this case, sentence) need not be a syntactic unit-- it can be any pattern stored in a special type of lexicon used exclusively for determining frame scope. The unit of scoping and its extent (e.g., "plus or minus *n*") can be determined independently for each frame class.

When a pattern that gives rise to a slot value of a type defined for a given frame class is found in the text, the slot is automatically mapped by FARR to any frame whose scope encompasses the location of the pattern. Thus, if the name of a corporation were to occur within the two sentence range of the tie-up frame in our example, it would appear in that frame. Of course, this may not be accurate-- DEFT has a tendency to overgenerate slots through bogus associations that arise because of this weak scoping mechanism.

Another issue that is encountered is overlapping frames. The "best available" resolution can be specified in the frame definition file. One alternative is simply to accept both frames, since they may be describing separate concepts. If the frames are of different classes, FARR supports the attribution of a priority to each class, and only the frame with the highest priority need be retained. If the frames are of the same class, FARR supports a "non-multiply occurring" attribute, which optionally suppresses all but one of the frames. Unfortunately, the action taken is generalized to all situations-- the specifics of a given case cannot be taken into account. Thus, DEFT tends to either overgenerate or lose frames.

When a message's frames have been generated and ambiguities resolved (to the extent that DEFT can resolve them), the frames (and the message) are routed to a destination directory on the basis of their content. Routing instructions are defined in a rule base using a normalized conjunctive form of field-value pairs. It should be kept in mind that although DEFT's primary mission is extraction, not dissemination, the routing capability (since it is based on knowledge representation) provides a sensitive mechanism for determining the destination of a message and the structured representation of its contents.

Controlling the System: DEFT Tools and Specification Management. In order to make DEFT portable to different computing environments and problem domains, the definition of user-modifiable system characteristics has been exported to a set of external specification files. These files govern the interface with the surrounding message handling system, the output data model, FDF configuration, and other "housekeeping" functions. Specification files are maintained using any convenient text editor.

The most important system specifications from the standpoint of the end-user are the lexicons and the frame routing rules. To facilitate lexicon development and maintenance, a lexicon editor is bundled with DEFT that provides a graphic user interface (under X/Motif) for interactively defining lexicons and entering/editing lexicon entries. Lexicons can also be created/updated from databases or external machine-readable files (e.g. gazetteers, corporate name lists) using a batch load protocol.

Like the lexicon editor, the routing rule manager provides a GUI for maintaining routing rules. It uses a spreadsheet metaphor to minimize the user's exposure to the potentially complex Boolean logic that the rules can involve. Menus of valid values and conditions tests are automatically provided.

Another important DEFT tool is frame review. DEFT was developed under the assumption that a user would always be in the loop; it was not intended to run autonomously. This package therefore supports simultaneous display of messages and the frames derived from them, providing highlights that show where slot values were extracted. Menus of valid values drawn from the lexicons assist the user in filling slots that were omitted by DEFT. Features for selectively deleting superfluous slots and frames are particularly important, since DEFT (like other pattern-based approaches to text analysis) tends to overgenerate data. A mechanism is also provided to facilitate manually linking frames of different classes into higher-level logical aggregations, since DEFT was not originally designed with an automated linking capability. Clearly, these two design assumptions-- human interaction and manual frame linking-- had an impact on working with the MUC-5 data.

DEFT as an Engineering Shell

This description of the DEFT system has emphasized that analysis threads are composed of independent components which communicate through a common data structure using a library of utilities that constitute an API. It is our contention that

DEFT's strengths are:

- A powerful pattern searching capability, which we are extending.

- The ability to integrate COTS, GOTS, and custom-written programs within the DEFT architecture.

We believe that there will probably not be a single text analysis or NLU system that meets the requirements of all conceivable applications. There will always be a tradeoff between such factors as speed, depth of analysis, breadth of coverage, portability, robustness, and analysis methodology that will favor one technology over another for a particular problem. The real question is not "What is the best system?", but "What is the best system at this moment?"

Our current development work on DEFT is chiefly targeted at its usefulness as an integration tool. DEFT provides a high-speed pattern searching capability which can successfully extract data from structured or tightly constrained textual inputs, while providing pre-processing services (e.g. tagging words with part of speech or semantic attributes) for third-party software which performs more extensive natural language processing for unconstrained textual inputs. This approach should be especially efficient for applications in which messages are mixed (formatted and unformatted), text analysis tasks are varied in complexity, and throughput is a major consideration.

Inherent Limitations in DEFT's Pattern-Matching Approach

Because DEFT was not originally intended for problems of the scope of MUC-5, its simplistic approach posed some major problems. Among the most fundamental were:

Syntactical Patterns. DEFT has very powerful mechanisms for specifying "atomic" patterns-- a corporate name, a place name, a set of words that indicate a joint venture, etc. DEFT was not designed to have the capability of expressing relationships among the patterns in its lexicons and providing for the assignment of values defined with respect to these patterns to variables. These are essential capabilities for the implementation of the most rudimentary semantic grammar. For example, DEFT had no way to express: "Look for a corporate name followed by a joint venture formation phrase and take the following corporate name as the partner in the joint venture."

Frame Scoping. DEFT was designed to interpret the scope of a frame as a function of proximity to the "hit location" of the pattern that resulted in a frame's instantiation. The boundaries are determined by a pre-defined number of repetitions of a pattern contained in a scope lexicon. An upper ceiling determined by a fixed number of characters can also be specified, in case the scoping pattern is not detected a "reasonable" distance from the site of the hit. All occurrences of slots defined for a frame within these boundaries are automatically included in the frame when it is assembled by FARR.

For highly formatted text (e.g. messages in Military Text Format), such a mechanism is adequate. For free-text, it is not. In the MUC-5 evaluation, DEFT failed to report valid objects that it located (notably entities) because they were not within the scope of a tie-up, as DEFT measured scope.

Frame Linking. The original DEFT design assumed that a human operator would perform this task. Automated linking is obviously needed for "unattended" operation and is clearly useful even if there is a human-in-the-loop.

Solutions

Current internal research and development work aimed at resolving each of these problems for the eventual DEFT product adheres to the constraint that architectural extensions must be philosophically compatible with the pattern-based approach, while avoiding significant overlap with NLU (which we prefer to view as an integratable component in a complex system). As noted earlier, key software being developed under IR&D was not available for the MUC-5 final evaluation; however, work continues and will be tested on the MUC-5 corpus in the near future to validate the approach.

Syntactical Patterns. This is the specific area that was not developed in time for the evaluation; unfortunately, it is also the most critical for dealing with even the simple aspects of the MUC problem. The approach we selected is intended to be compatible with the integration of more powerful text understanding components in the future, while extending the range of problems DEFT can solve by itself. It exploits DEFT's atomic pattern-recognition capabilities while separating the definition of a semantic grammar into an independent extraction phase. This phase could easily be replaced (or supplemented) with an NLU system which can optionally take advantage of the DEFT lexical pre-processing while performing deep syntactic and semantic analyses. This separation is in part intended to provide an initial test of our belief that the integration of DEFT with an NLU component creates a symbiotic association with better performance characteristics than either system by itself.

To stay within the (admittedly loosely defined) bounds of pattern matching, our approach to exploiting syntax consists of providing DEFT with a simple mechanism for expressing "meta-patterns"-- that is, patterns whose components may be the atomic patterns (and, by reference, their attributes) located by the DEFT lexicons. We decided to use a BNF specification to define a semantic grammar based on a combination of literal strings and DEFT-identified tokens.

The key issue was how to pass the results of DEFT pattern-matching to the parser. An integrated NLU component within the DEFT shell could interface directly with the DEFT Tag File through the API; the component could also interface with the frames generated by DEFT, providing a preliminary level of analysis on which to build. For our prototype, however, we chose to mark terms in the text with SGML-like tags to indicate their properties. The grammar directly references these tags, and routines were provided within the parser for assigning text strings to slots by extracting DEFT lexicon attributes (e.g. normalized values or semantic characteristics) or collecting words intervening between two tags (of the same or different class). Additional primitives for manipulating the strings prior to slot assignment were also built into the parser infrastructure to control frame generation and the assignment of slots (including pointers to other frames) to frames. This significantly improves on the primitive scoping capability provided with the current version of DEFT.

The approach selected thus provides a vocabulary for expressing both the expected contents of documents and the rules for instantiating and linking templates. At the same time, its intermediate product is human-readable (and, in fact, could be used as a general-purpose "tagger") and easily interpreted by other programs.

Frame Scoping. Fundamental changes in the DEFT frame-scoping mechanism are planned which will exploit domain knowledge as well as limited syntactic (from the meta-patterns) and semantic (from lexicon attributes) data. For MUC-5, the basic DEFT mechanism was retained, with its inherent limitations.

Frame Linking. A primitive frame linking capability was added to DEFT. It was based on frame scoping, however, and therefore suffered from the same limitations. The DEFT frame definition file format was extended to accommodate hierarchical relationships; any frame defined as a child of another frame had its generated frame ID automatically included as a slot value in the parent frame if its "hit location" fell within the scope of the parent frame. Of course, multiple and spurious associations are easily generated in this way. In the future, frame linking will be improved by combining syntactic and domain knowledge in a final extraction phase to resolve inter-object relations.

RESULTS

The results of the final MUC evaluation were strongly influenced by the unavailability of the parser, which was an essential component of the DEFT approach to MUC-5. The resulting scores indicate the magnitude of the problems inherent in a simple pattern-matching strategy which is not informed with even a crude semantic grammar. It should be noted that a decision was made to focus only on a subset of templates and slots required for the preliminary run. These were the document template, tie-up-relationship, and entity. The F-measures for the final evaluation were:

P&R	2P&R	P&2R
1.15	2.64	0.74

Not surprisingly, these were the lowest scores for any system in the evaluation. A detailed analysis of the run is of little utility, however there are some points of interest seen in the walk-through sample document.

Walkthrough Document

The identifying data (document number, source, and date) were correctly extracted.

Some simple atomic patterns were defined in a DEFT lexicon for tie-up relations. These were to be factored into a semantic grammar; as noted, the parser was not available at the time of the run. Therefore, the patterns were run as a simple search. It correctly identified the presence of a joint venture in the sample document, incorrectly instantiating two tie-up templates (one for each of two out of three references to the venture) and entering their IDs in the content slot of the document template. DEFT currently does not determine that multiple references have a common object unless the frames overlap.

A single entity was mis-identified, "Jiji Press Ltd.," which is actually the document source. This entity was incorrectly associated with the first tie-up. The foregoing explanation of the DEFT scoping mechanism makes it clear why this false association took place. The name of the "BRIDGESTONE SPORTS CO." was correctly reconstructed from the corporate designator ("CO.") and assigned to the first tie-up. The name of the joint venture, "BRIDGESTONE SPORTS TAIWAN CO.," was also constructed and associated with the second tie-up instance. No other features were correctly identified.

Among the other corporate names, the algorithm used by DEFT would not have identified "UNION PRECISON CASTING CO.," but did identify "TAGA CO." However, this entity was considered out of scope of the tie-up templates and was (incorrectly) not attached to one. DEFT had no facility for recognizing "BRIDGESTONE SPORTS" nor for tracking the reference to "THE NEW COMPANY."

What Worked

DEFT was effective at recognizing literal strings and patterns contained in its lexicons. DEFT frequently generated correct entity names that were not in the corporate name lexicon using a set of heuristics that reasoned backwards from a designator. For example, "BRIDGESTONE SPORTS CO." was constructed. DEFT of course had little problem with the tagged items for the document template. These are precisely the kinds of elemental functions that DEFT is expected to perform well.

DEFT recognized the occurrence of some of the joint ventures, based on a very limited set of patterns that were originally defined for use in connection with a semantic grammar. This set could have been extended to produce improved recall had we known the parser would not be available. These few successes indicate that even a simple pattern-based approach can recognize concepts of this type in restricted cases.

What Failed

The lexicons and extraction phases that were rapidly developed for MUC-5 contained some bugs that were not observed during training; some corporate names were missed, for example, that should have been constructed. The chief failings were inadequate lexicons for identifying joint ventures and inadequate scoping. These two problems combined to suppress the instantiation of the many valid entities that DEFT found, but could not associate with a tie-up relation and therefore did not report. In general, the system was configured to reduce the anticipated overgeneration, with the expectation that tie-ups and entity relations would be identified and scoped by the semantic grammar; in the absence of the parser, undergeneration became severe.

System Training and Resources Expended

The effort expended on MUC-5 testing and documentation was approximately two person-weeks. System development activities undertaken independently of MUC-5 were exploited for the MUC-5 evaluation run. These included:

- Analysis: 1 person-month
- Lexicon Development and Data Definitions: 1.25 person-months
- Extraction Phases and Functions: 3 person-months

The total level of effort for all activities impacting MUC-5 was therefore roughly 5.5 person-months.

As we have noted, key system components were ultimately unavailable for the MUC-5 evaluation. Although we won't know "how we would have done" until the components are completed and our internal tests against the MUC data are repeated, it is our expectation that significant improvement will be obtained with a little additional effort-- although performance is neither expected nor required to approach that of true NLU systems, given our view of DEFT as an integration environment.

Most of the effort in creating a new DEFT application usually centers on lexicon development. For MUC-5, most lexicons were batch loaded from the data supplied via the Consortium for Lexical Research. A few lexicons for joint venture identification and scoping were developed manually. These were quite simple and their actual creation required minimal time.

Much of the time on MUC-5 was occupied with writing C-code for extraction routines, particularly for corporate names. The need to write so much code for a new application is a current weakness in DEFT which will be remedied to a degree when the parser becomes available.

Of course, a key activity was the analysis of the test corpus and development of a semantic grammar appropriate to the EJV problem. The results of this analysis were manifested in the tie-up relation lexicon and the BNF grammar for the parser. Only the former was ready in time for the evaluation. Analysis was a cyclical, iterative process; refinement continued during system training.

DEFT system training consisted of a series of runs against samples of the training corpus, utilizing the frame review tool to examine the results. Lexicons were manually refined as a result of missed objects and false hits. Early runs resulted in changes to the batch loading sequence for some of the lexicons (e.g. the corporate designators). Feedback into the grammar would also have been derived from this process, had the parser been available and time permitted. As it was, time was insufficient even for lexicon refinement; for example, a few key errors in the corporate designator lexicon resulting from a bug in the program that prepared the file provided through the Consortium for batch uploading were noted only after the final evaluation run was analyzed. This was partially responsible for some of the undergeneration.

What We Learned

It came as no surprise that simple patterns are inadequate to extract the complex ideas expressed in the EJV documents. We view the results as validating the concept that DEFT, operating as a standalone system, is best qualified to perform on problems involving well-defined, constrained sets of text objects to be extracted, even with the addition of a "meta-pattern" or grammatical capability. DEFT should excel on such problems when throughput is a major consideration.

The selection (and on-going implementation) of a mechanism for expressing meta-patterns that is compatible with all of the goals discussed earlier is a major outcome of our MUC work, even though it was not available in time. We believe that this approach will significantly empower DEFT and broaden the range of applications for which it is a suitable tool, while increasing the flexibility with which it can be integrated with other text analysis tools. This will prove highly valuable to our current government customers, as well as future DEFT users in the government or commercial sector.

DEFT's potential as an integration environment was underscored by the fact that we successfully ran documents through:

- A complex set of extraction phases
- With extremely large lexicons

that are beyond the scope of anything that has been tried in existing DEFT applications. The robustness of the architecture and efficiency of the pattern searches were our major consolation in the MUC-5 evaluation. We therefore look for opportunities to combine DEFT's system engineering and search capabilities with the sophisticated analytical power of NLU-based solutions when real-world problems are encountered which are out of scope of DEFT's simple extraction mechanisms.