DESCRIPTION OF THE SAIC DX SYSTEM AS USED FOR MUC-6

Lance A. Miller

Science Applications International Corporation
8301 Greensboro Drive, MS E72
McLean, Virginia 22102

lamiller@cpva.saic.com

(703) 556-7079

BACKGROUND

This is a very young project, operational for only a few months. The focus of the effort is *data-extraction*, the identification of instances of *data-classes* in "commercial" text -- e.g., newspapers, technical reports, business correspondence, intelligence briefs. Instances of data-classes in text are phrases which identify factual content, such as names of people or organizations, products, financial amounts, quantities, and so forth.[1]

A number of applications would be very well served simply with automated and accurate identification of the data-classes that occurred in their texts of interest, with interpretations left to experts. Such applications include extraction of bibliographic information, document indexing, competitive analyses based on open sources. technical information retrieval, foreign technology and political assessments, tracking financial and other resource transactions in the written media, and various types of link analyses based on text correlations.

Providing very accurate automated data-extraction capability for this type of application is the objective of the DX project. Relative to the broad language understanding goals of many projects, this effort is clearly a niche one, but perhaps all the more ambitious in its accuracy goals for that reason. Last October, 1994, we started designing a new pattern-recognition language to be developed for just the purpose of data-extraction. In May, 1995, the first version of the language, DXL (for "Data eXtraction

[1] In our view, data-classes comprise IDs and "quantifications". IDs are either proper names or technical descriptions (of a "part-of" or "kind-of" type). Quantifications are either measurements or mathematical or logical expressions.

Language"), was available and training begun, but because of personnel unavailability, serious use of it did not begin until September.

Other elements of the DX system are similarly quite new. The underlying text-processing system was adapted from an ongoing SAIC-Navy project to convert legacy technical manuals into a particular data-base form suitable for interactive electronic use on a portable computer.[2] Numerous modifications have been made to this system including a new data-base access capability developed in September for rapid access to many millions of words and facts. The fact contents of the data-base (called the knowledge-bank), such as names of places, persons, and organizations, were acquired only in the last few months and entered using a coding scheme which represents their many syntactic and semantic features. Also recent is the accumulation of the large number of critical (for our approach) "signal-words" which mark the presence of certain data-classes -- for example, "Mr.", "Ave.", and "Corp." suggest the nearby occurrence of person-names, street-names, and organization-names, respectively. Similarly, the needed linguistic infrastructure has also only recently been developed: the compilation from several sources of a large lexicon with parts-of-speech, the development of the capability to find the "roots" of words prior to lexical look-up (e.g., the root of "placing" is "place"), and the morphological analysis capability to permit guessing at the part-of-speech of unknown words based on their suffixes.

The life-cycle of the DX project is thus independent of the MUC, which has been a very useful, if not the best-timed, experience.

SYSTEM DESCRIPTION

In this section we summarize the elements of our design objectives and provide extensive details of the general system implementation.

Design Approach

Two sets of considerations determined our selected approach to the data-extraction problem: our set of performance objectives, and the results of our analyses of data-classes. Concerning performance, we wanted a system that would most of all be highly accurate -- less than 5% misses and false alarms -- yet robust in the sense of failing seldom, failing soft, and restoring easily. We also wanted a system that was easily extendible, serviceable by programmers, supportive of informed guessing (but giving confidence and basis), and eventually capable of learning extensions. Extremely rapid processing is not a requirement, in the belief that achieving the quality goals first could be followed subsequent speed-up enhancements (e.g., parallelization).

Our analysis of a wide variety of data-classes indicates that the majority of classes, and of instances, do not require sentence-level linguistic information for their detection and identification. That is, it appears that having a good sentence parse would only infrequently be of value in determining the data-classes embedded in the sentences. The needed information seems to be primarily available within the data-class phrase itself, in the form of obligatory and optional elements. For the majority of "difficult" cases, it would appear that the semantic features of local adjacent contexts plus some global context (e.g., the type of document) could resolve the identifications.

These considerations motivated the three-stage strategy we adopted for the DX project. The first is unabashedly a brute-force method and is to **identify by look-up**: for those data-classes like person-

[2] This conversion activity is part of the Navy program to develop the support technologies for producing and using Interactive Electronic Technical Manuals (IETM) for new and existing systems, as governed by the MIL-M-87268 standard. In turn, DXL is now being used as the specification language for recognizing various components of existing Navy tech manuals in the SAIC IETM project.

names and organizations that are amenable to this approach, get as many instances as possible and enter in the knowledge-bank and then check for them in any new input. Secondly, **identify by pattern**, using a language specially developed to have all the functionality useful for this approach, especially very powerful pattern-matching capability coupled with the facility of placing arbitrary constraints on the patterns or local contexts. The final step is to **identify using semantics-based contextual reasoning**, wherein potential remaining data-class targets are fuzzily classified as one of several related data-classes (e.g., proper names) on the basis of suggestive cues (e.g., capitalization). Then, reasoning heuristics (e.g., "names of people often have appositives whose heads or modifiers are marked semantically as being characteristic of people") are used to direct the search for discriminating contextual clues.

This three-fold approach can be argued as meeting most of the performance goals and certainly fits the finding of data-classes being pattern-based. However, it is an empirical question whether the desired accuracy levels -- particularly <5% misses -- can truly be achieved without at least a reasonable identification of the case-roles of the surface constituents. For this reason, an additional design requirement for the new pattern language was that it would have the capability to represent very powerful sentence-level parsing grammars (e.g., context-sensitive rules, unrestricted look-ahead, easy representation of discontinuous constituents).

Implementation Features

System-level Overview.

A much-simplified view of the DX system is given in Figure 1. The first step is to convert the character-stream input into a stream of "tokens", essentially words, based primarily on the presence of blanks between character-strings. Upon identification, each token is annotated with a set of "prime" attribute-values including: the most important attribute "type" ("word", "num", "sgml", "punc", or "mix"), start/stop character positions, "chars" (the token's character string), capitalization ("capital", "upper", "lower", or "mixed"), and token number. The value of the "chars" attribute is that string which has all beginning brackets (or parentheses or braces) removed as well as all ending brackets and punctuation of all kinds; appropriate attributes are set to indicate what was removed. In addition, the singular possessive marking " 's " is also removed and a possessive attribute set. In this way, the token "chars" value can be looked up in the knowledge-bank without worrying about punctuation of any kind.

All of the subsequent system processing is accomplished with specific pattern-rules and involves moving up and down the token stream, checking tokens for particular attribute values, and, when successful, either changing the attribute-values of single tokens or replacing several tokens with a new one (with a new "type" attribute, whose "chars" attribute is the concatenation of those replaced). For any particular rule, the whole document is searched from the first token to the last.

The second major processing step is to check the knowledge-bank to see whether the "chars" value of tokens are known as an instance of a data-class of interest (e.g., known locations or organizations). If so, the tokens corresponding to the known data-class entry are replaced with a single token whose "type" is set to the name of the data-class. In addition, new attributes are added, including part-of-speech, the word-stem, number (singular or plural). Token "chars" not recognized as data-classes are analyzed morphologically to determine their part-of-speech, word-stem, and number.

In the third step, the major data-class recognition rules are applied. The knowledge-bank is again a major source of information but this time primarily for "signal-words" which usually preface or terminate data-class patterns. For example, a number of organizations fit the canonical form of a "prefix organizational title" followed by "of" or "for" followed by a location or unknown capitalized words (e.g. "Bank of New York"). Prefix titles like "bank" are characterized in the knowledge-bank by a number of

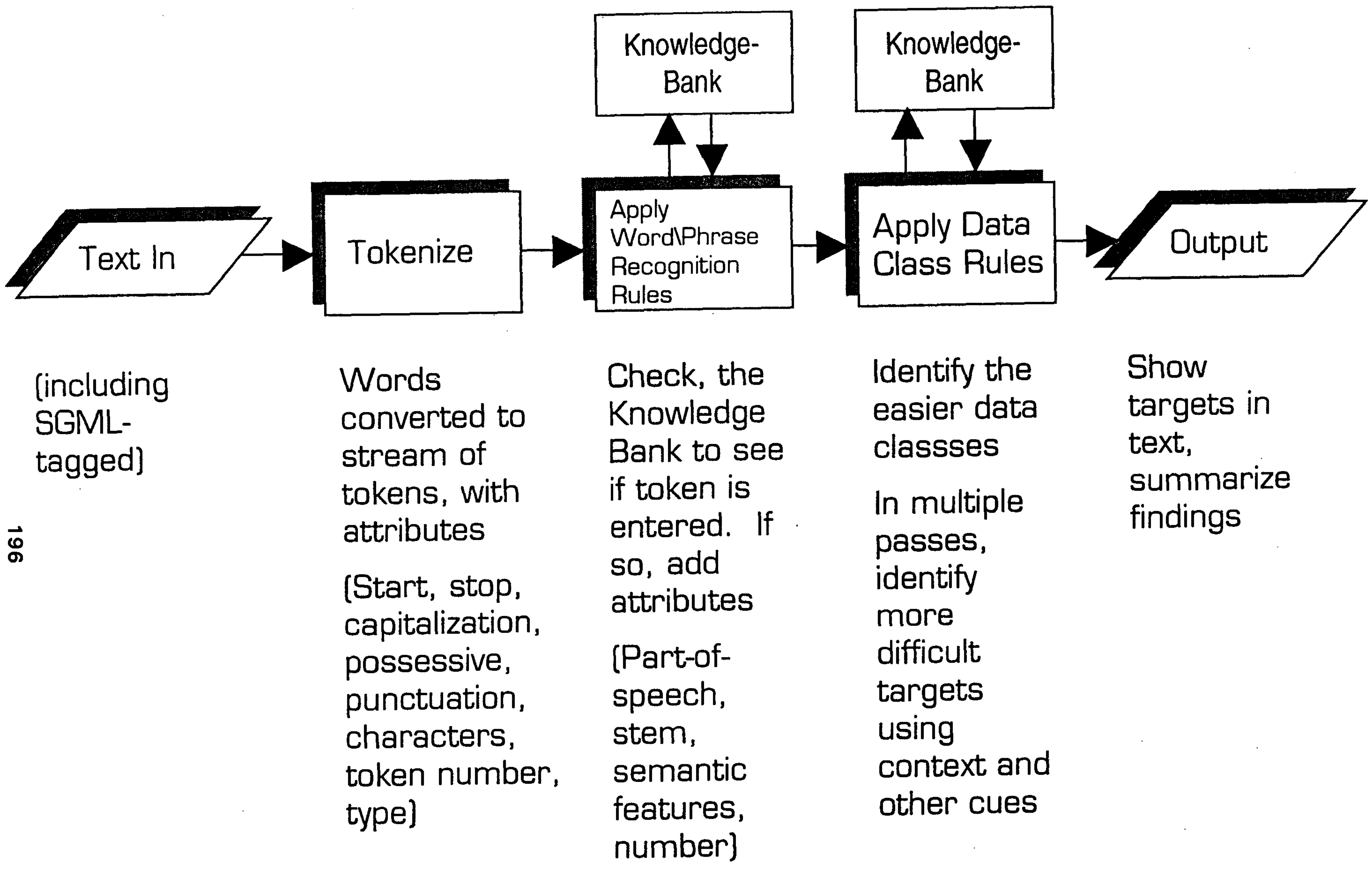
Text In
(including SGML-tagged)
Tokenize
Words converted to stream of tokens, with attributes
(Start, stop, capitalization, possessive, punctuation, characters, token number, type)
Knowledge-Bank
Apply Word\Phrase Recognition Rules
Check, the Knowledge Bank to see if token is entered. If so, add attributes
(Part-of-speech, stem, semantic features, number)
Knowledge-Bank
Apply Data Class Rules
Identify the easier data classses
In multiple passes, identify more difficult targets using context and other cues
Output
Show targets in text, summarize findings

Figure 1. System Overview

attributes, such as being capitalized, not a name, referring to an organization, whose activity is commercial. A token whose "chars" value is "bank" will not have these attribute-values already associated with it. Rather, only when some rule asks whether the token has the attribute values associated with an organizational title will the knowledge-bank be checked for these. As a result of checking for such attributes, the values returned will be set on the token. This process of adding values to a token only when a rule has inquired of the attributes is known as "lazy annotation" and is much more economical than attaching all possible attribute-values slavishly to all known knowledge-bank entries.

An example of these processing steps is given in Figure 2 for the input "Dr. Joyce Smith lost money." The tokenization and associated attribute-annotation for the first two words is shown as lists of attribute-value pairs in the LISP-list format used by the Scheme programming language in which the system is written (supplemented by a few C and C++ modules). The next line on Figure 2 shows what the results would be were the knowledge-bank looked-up for instances of various kinds of titles.[3] In this case the character-string "dr" is associated with two types of titles, one a prefix for person-names and one a suffix for in-city street references; both values are returned for the attribute **title_typ**. For a person-name recognition rule such as is shown in Figure 3 below, the likely result would be the replacement of the first three tokens ("dr", "joyce", and "smith") with a new single token of **type** "person", with start/stop values reflecting the span of coverage in the input, and with a new attribute giving the name of the rule that was successfully applied ("cap-person"). Outputs include both a tagging of the target in the input and the inclusion of the target characters in a list of other such person targets.

The Data-Extraction Language, DXL

Five of the major features of DXL are: (1) extremely flexible pattern specification, (2) unrestricted rewrite capability, (3) the capability to put constraints on pattern elements or to invoke global constraints, (4) the capability to invoke, anywhere, the full power of the Scheme programming language, and (5) the ability to expand complex "chars" strings.

The first feature is illustrated by Figure 3, which shows a DXL rule for one of the PERSON canonical forms. DXL rules have three components: a left-hand-side (LHS) specifying the total pattern that is to be matched, a right-pointing arrow indicating that the LHS is to be rewritten, and a right-hand-side (RHS) indicating what is to be substituted for the LHS, with what actions. In Figure 2, the LHS has four elements: an optional pattern followed by an obligatory one, followed by two optional patterns. This rule would match instances such as "Dr. Harry Morgan Raffler, Jr., Vice President" and "Frank".[4] The meaning of the prefix symbols is given in the following table:

*	An optional operator, zero or more occurrences, match the minimum
?	An optional operator, zero or one occurrence, match the minimum
+	Obligatory element, at least one, possibly more
!	When placed after the above, forces a match of the *maximum* number of specified patterns
@	Indicates a defined pattern

The LHS therefore involves four defined patterns: 0 or more prefix titles for people (such as "Prof."), at least one capitalized word (with no comma following it), an optional family title such as "Jr.", and an optional suffix title, such as "Director ". Two of the four BNF pattern definitions are given at the bottom of the Figure. That for **title_pref**, for example, specifies that the "chars" value should have an initial capitalized letter, that the type of the title ought to be "pref" (for prefix), and that the value of **c1**, the second hierarchical knowledge classification value (**c0** is the first), should be "person".

[3] This would not actually be a good way of approaching recognition of person-name instances, but it does illustrate what additional attributes would be added as a result of consulting the knowledge-bank.

[4] Such a rule is only illustrative and would actually *not* be used because so many optional elements beg for over-generalizations.

Input:	"Dr. Joyce Smith lost money."
Tokenization	((chars . "dr")(endpunc . ".")(wordcase . "capital")(start . "1") (stop . "4")(type . "word"))
	((chars . "joyce")(wordcase . "capital"))start . "5"))(stop . "9") (type . "word))
Kbk Lookup	((title_typ ."p-pref""street_suf")(chars ."dr")...(type . "word"))
Apply Rule	((type . "person")(rule . "cap-person")(chars . "dr joyce smith")...(start . "1") (stop . "16"))
Output	(1) Dr.<person>Joyce Smith</person> lost money.
	(2) Person_File
	Dr. Joyce Smith

Figure 2. Example of Procesing

```
   optional          obligatory      optional       optional
 *!(@title_pref)     +!@capword      ?@title_fam    ?@title_suf

     ==>     person [(seta! 'rule "cap-person")
                     (seta! 'conf "90")
                       ...
                     ]
Where:

@title_pref :=  wordcase:"capital" & title_typ:"pref" & c1:"person"
@capword :=  wordcase:"capital" & endpunc:~","

...
```

Figure 3. Example of a DXL Rule: Person-name

Discontinuous constituents are easily specified using the "*" (Kleene star) operator, as in

@first_pattern *.any @second_pattern

where ".**any**" means a token with any "type" value, and the star indicates zero or arbitrarily many intervening tokens between the two patterns of interest. To rewrite these into new patterns requires use of the dynamic variable-assignment facility which takes whatever tokens were matched by the pattern and assigns them, as a list, to the variable indicated, as shown by

@first_pattern ->v1 *.any->v2 @second_pattern ->v3

==> $v1 [actions] $v2 $v3 [actions]

On the LHS, variables v1-v3 are assigned to the three pattern elements; those binding to v1 and v3 should, in this example, be understood as binding to single tokens, and these are referenced on the RHS, via the "$" operator. Single tokens bound to v1 and v3 are modified by attribute-changing actions, and all the bound tokens are finally reinserted back where they were in the token stream.

The second, rewrite, feature of DXL is illustrated by the abstract rule below, which, for clarity's sake, omits the needed variable-assignments:

\A\ B [X] C D {Y} /E/ ==> C F B

This rule specifies a left-context of A after which three obligatory patterns are sought, **B-D**, where **B** has some additional condition X placed on it. There is also a global test **Y** which has to succeed, all this in the right context of **E**. If all this LHS succeeds, then **B** and **C** are to be permuted, **D** is to be deleted, and **F** is to be added (inserted between C and **B**), with neither A nor E being further involved. With LHS context-sensitivity and the ability to permute, add, and delete LHS elements -- and also unrestricted look-ahead -- the rewrite power is essentially unconstrained, beyond recursively-enumerable context-sensitive grammars, having the power of a Turing machine.

Local and global constraints (the third feature) have been illustrated, and they are often expressed in practice by dropping into full Scheme code (the fourth feature). The fifth feature, expansion is very useful when the "chars" attribute is a mixture of letters/numbers/symbols. Expansion permits the components of a complex "chars" string to be broken apart and analyzed using the same rule formalisms employed for multiple tokens. For example, the following rule expands all tokens of "type" "mix":

type:"mix" ->v1 ==> (expand v1)

The result of expanding a token whose "chars" are "45lbs./sq.in.", is to replace this token in the token stream with the following ones:

.sot 45 lbs. / sq. in. .eot

The first and last tokens have special type values, "start of (expanded) token" and "end of (expanded) token", but they have no start or stop attributes, being "dimensionless" with respect to characters. The inserted tokens can now be recognized as a *measure* by a rules such as the following:

.sot type:"num" @unit "/" @unit .eot ==> measure

where the pattern **@unit** is appropriately defined as a single or multi-word reference to a unit of measure.

The Knowledge-Bank

The knowledge-bank has five major types of entries (with the approximate quantities presently being added given in parentheses): (1) words or phrases which are given a part-of-speech and perhaps a word-stem attribute, but little else (64K); (2) words/phrases which are full instances of data-classes (such as person first-names, cities, organizations) (6M); (3) "signal-words", usually called "titles" which indicate the likely presence of data-classes (such as "Ave.", "p.m.", "Bank", "Mlle.") (1K); (4) "clusters" of ordinary words which share significant semantic features (such as "communication acts") (2k); and (5) isolated ordinary words which have particular significance of one kind or another (1K).

The first type of entry, mostly common words, facilitates sentence parsing when needed. The second type implement the brute-force "identify by look-up" principle of the DX system. The third type contribute the most in supporting the pattern-based identification principle of the system, as facilitated by the fourth and fifth types.

The last four types have a part-of-speech attribute plus several classification attributes plus a number of other features. It is this rich set of features that are used as conditions on the DXL pattern elements and largely underlie the potential for very high accuracy in target detection. A sampling of knowledge-bank entries illustrating these features is given in Table 1.

In the first row of Table 1 the first "key" column indicates the lower-case look-up characters that will be matched against the "chars" values from the input text. The rest of the column headings in the first row are various attributes which are relevant to the examples given.[5] The entry for "susan" shows that its highest knowledge-classification feature, **c0**, is "hument" ("human entity"), sub-categorized by the next classification feature, **c1**, as "person", in contrast to the second entry, "ibm", which is an organization. These two entries also are coded as being names and capitalized. The third entry, "mr" is also categorized as referring to a person and capitalized but is not a name, rather a title, of type "pref" (for "prefix"). The fourth entry, "militant", again refers to a person, but is neither a name nor a title (nor capitalized) but has a **role** value of either "political" or "terrorist". The string "texas" is identified as a place, sub-category "center" and sub-sub-category "state", and it is both a name and capitalized. The last two examples both have the knowledge-classification features **meas-time-date**, but "christmas" is also categorized as a "holiday" and is both capitalized and a name; the phrase "paid holiday" is neither capitalized nor a name but fills the semantic role of referring to a "holiday" entity.

These examples suffice to illustrate the rich set of syntactic and semantic knowledge-bank features that are available for use in the DXL rules to identify and discriminate among even very similar data-classes.[6] These features would also provide the basis for a neural-net or deterministic classification approach (e.g., C4.5) for a learning capability to be developed later.[7]

SYSTEM PERFORMANCE

[5] The Table 1 attribute-by-entry structure is convenient for exposition, but given there are now well over a hundred total possible attributes, such a structure would be very space-inefficient. The knowledge-bank actually has a much different representation. Also, not all relevant attributes are shown. For example "susan" also has a **name_type** attribute with the multiple values of "given | female".
[6] For example, the existing features can discriminate among five types of capitalized references relating to people, as indicated by the following: "reagan", "american", "christian", "irish", and "mr. president".
[7] We envision examples of new data-classes first being clustered by hand into high-similarity groups, but this may not be necessary.

Key	c0	c1	cap	name	title_typ	role	c2	c3
susan	hument	person	1	1	-	-	-	-
ibm	hument	org	1	1	-	-	-	-
mr	hument	person	1	0	pref	-	-	-
militant	hument	person	0	0	-	polit/ter	-	-
texas	place	center	1	1	-	-	state	-
christmas	meas	time	1	1	-	-	date	holiday
paid holiday	meas	time	0	0	-	holiday	date	

Table 1. Example Excerpts from Knowledge Bank

Sequence of Processing Used in the Named Entity Task

Nine major processing steps were employed for identification of the data-classes involved in the MUC-6 Named Entity task., as described in the following table.

No.	Step	Description
1	Multi-Word Lookup	Identify the data-class of multi-word phrases found in the knowledge-bank
2	Find-Root	Check to see if remaining words have a part-of-speech by direct look-up first and then by stripping plural suffixes
3	Morphology	Guess at unknown words' part-of-speech based on word suffixes. Add number (and gender, person) attributes where appropriate.
4	Single-word Places	Identify all instances of single-word **places** (such as "Chicago").
5	Level 1 Data-Classes	Apply DXL rules for data-classes which are non-interacting and do not require other data-classes for their identification (also parallelizeable). These were: **time, date, percent, money** (some references to **time** involved single-word places, e.g., "4 p.m. Chicago Time").
6	Level 2 Data-Classes	Apply DXL rules for more complex interacting rule-sets in the following order: **place1, place2, org1, org2, person1, person2, person3.** The numeric suffix on the rule-sets indicates that data-class rules were clustered into groups of canonical forms that were increasingly complex, as indicated by the number-value.
7	Last-Pass	A set of DXL rules which checked the token stream for any remaining tokens which could be part of **place, org,** or **person.**
8	Adjust	A set of DXL rules which adjusted the start/stop positions of the targets to include or exclude punctuation according to MUC-6 rules.
9	Tag	Insert the appropriate SGML tags around the targets in the input text as calculated by the Adjust rules.

In developing the rules for a particular data-class, the following five-step strategy was employed: (1) a large number of examples of the data-class were collected, and these were clustered into groups having high internal similarity; (2) a canonical pattern-sequence of obligatory and optional elements (plus signifying contexts) was developed for each cluster; (3) DXL rules were developed for each canonical form; (4) the rules were ordered in a sequence which causes those with the most number of fixed patterns to be run before those with fewer, and in the case of a tie, with those accounting for the largest number of instances to be run first; and (5) the rules are broken into two groups with the simpler reliable ones grouped in one rule-set, e.g., **place1** for the location rules, with the complex rules placed in a second rule-set, e.g., **place2**. Some experimentation was performed to determine the final grouping of the rule-sets into the Level 1 and Level 2 sets, but that indicated in the table led to the best results.

In the Last-Pass step, there were several rules which attempted to use context-inferences and other heuristics to identify token sequences which were likely **place, org,** or **person** instances. One such was to see whether a promising token (a capitalized unaccounted-for one, for example) was an element of any of the instances of the three types of data-classes that had previously been identified or was an acronym thereof. Thus, both "Consuela Washington" and "Ms. Washington" were recognized by straight-forward person rules (the first by one which looks for known first-names; the second by one which uses prefix titles). The subsequent reference simply to "Washington" was correctly identified by the previously-seen by being a sub-string of, in this case, the closest prior reference, that of "Ms. Washington".

Development Effort

The DX project team for the MUC-6 participation involved seven people, most becoming involved recently. Team participation and responsibilities are shown in the following table.

Person	Organization	Responsibility	Period of Participation	Ave. % of Time Devoted to Project	Approx. No. of Person-Months
Cara Clouston	SAIC	Knowledge-Bank and System Administration	9/1/95 - present	85%	2
Tom Hicks	Outside Consultant	DXL design and implementation, overall system development	10/17/94 - present	85%	10
Tom Joyce	SAIC	Collection of data-class examples, coordination of MUC materials	8/1/95 - 10/5/95	50%	1
Rodger Knaus	Outside Consultant	DXL data-class rule writing	5/10/95 - 10/5/95	60%	3
Lance Miller	SAIC	PI, DXL design, Knowledge-Bank design, limited rule writing	10/1/94 - present	35%	5
Sarah Officer	SAIC	DXL implementation, limited data-class rule writing	10/17/94 - present	30%	4
Afsar Parhizga	Outside Consultant	DXL data-class rule writing	9/4/95 - present	100%	3

Roughly a third of the 28 person-month effort was devoted to design and implementation of the data-extraction language, DXL; another third went for overall system and knowledge-bank development; and the last third was focused on development of general and MUC-specific data-class recognition rules using DXL. Not counting the peculiarities of MUC requirements for tagging the identified data-classes, <u>no</u> part of this effort has been spent on non-reusable MUC-specific activities.

There were a number of factors which made the timing of the MUC-6 contest inconvenient relative to the external factors determining the pacing of development of the DX system:

- The knowledge-bank, upon which all processing relies, was designed and populated only in the August-September period; its implementation needed to be radically modified in September to permit handling of massive numbers of entries, and it is still not yet fully reliable
- The rule writers and knowledge-base/system administrator were effectively not available until early September
- The language, DXL, while possessing all the needed functionality, had limited high-level function libraries which are only being developed gradually with experience in rule-writing
- The system processing stages (esp. tokenization) are still under development, and the system still has quite limited debugging facilities

- The late release of MUC-6 task information and materials in August precluded advance study of the complex scoring, recognition, and tagging criteria

Training

The primary method of obtaining training materials was to extract a very large number of instances of each of the seven Named Entity data-classes from the provided test materials and use these files for development of the canonical groups and rules. An enormous amount of time was spent learning how to use DXL (and, as the rule writers did not know UNIX, learning the Linux system). An equal amount of time was spent in learning effective strategies for writing reliable rules -- especially, learning to avoid the temptation of trying to recognize too many variations of data-class instances with a single rule. It is only in the past month, after the contest, that the grammar writers have learned the "right" level of ambition for writing a rule, testing it with the limited debugging capabilities, and revising it modestly.

We note that the Wall Street Journal style guide was very useful in reducing the number of training examples needed.

Performance on the October 5 MUC-6 Tests

We did not do well. It did not magically all come together at the last moment. Our three-week flurry of rule-writing simply didn't cope.

In addition to the developmental pressures noted above, the difficulties of learning to program in a completely new language in a few weeks, and the agonies of understanding MUC identification and tagging criteria, we also made the mistake of attempting a much too complex approach to identification of the proper name classes: using a "fuzzy logic" approach in which capitalized words had fuzzy membership in the three person/location/organization classes which were gradually and simultaneously to be resolved. It didn't work.

Performance One-Month Later!

We started over, and performance now is very high. On a recent test involving over a hundred test-cases per class, we logged on average 2% misses, 6% correctly identified but incorrectly tagged, and 0 false alarms for the four simpler classes of time, date, percentage, and money. For locations and organizations, the numbers are 4%, 9%, and 3 respectively; the person rules are still in flux but will be in this ballpark by the time of the conference.

Things That Didn't Work

Aside from the fuzzy-logic approach, it took us a long time (several weeks at least) to learn that, for pattern-based rules, *simple* is terribly much more effective than *comprehensive*. We also suffered from the fact that, since the language DXL has only just been developed it quite reasonably still has a few bugs; these just happened to be difficult ones: e.g., coding that worked perfectly well in the main patterns of a rule LHS, did not do so in the left-context or in the pattern definitions (e.g., "@title_suf); the knowledge-bank could get corrupted just a bit without failing in the main; etc. And we had endless problems with the fact that the knowledge-bank is incomplete: the fact that it has so very many entries usually meant that problems were with our rules, not with the knowledge-bank, but there were many times when expected entries were simply not there or had been miscoded in some way. Finally, the absence of user function libraries to provide very high-level scotch-guarded functions for easy use in the rules, meant that too often we had to use very low-level programming functions or drop into Scheme, neither a forte' of the rule-writers.

Concerning specific target difficulties, we perhaps had the most trouble with organizations. Single-word organizations or ones without a prefix or suffix title (e.g., "Pilsbury", "Birds Eye") required context-sensitive semantics to pull in. Names with commas in the middle, commonly law firms (e.g., L. F. Rotchieff, Unterberg, Towbin), were difficult because we used the commas to suggest phrase boundaries. And organizations with "and" (in contrast to "&") as part of their name were difficult to discriminate from conjoined organizations (e.g., "Hollis and Pergamon Holdings, Ltd.", where "Hollis" is a reference to a prior-mentioned company).

Person targets were often difficult, but they tended to be the default case: we had already done our best with our second set of place and organization rules, and what remained was most likely a person. The high frequency presence of appositives with person-names provide one powerful source of semantic context resolution.

Things That Worked Well

The knowledge-classification feature hierarchy works termendously well in supporting identification and discrimination of data-classes (e.g., people having the main branches of HUMENT-PERSON, while cities have the main branches of PLACE-CENTER-CITY, and signal words associated with time references have the branches of MEAS-TIME-DATE-HOUR). The DXL language, now that we understand it, is wonderfully powerful and flexible. And the brute-force look-up approach handles over 40% of our instance recognition and is easily extensible. We are also quite encouraged by the success of some of the LastPass stray-pickup rules based on semantic contexts and other heuristics.

The Thing That Most Needs Reworking

The leading candidate for this prize is the user function library, to keep the rule-writer out of the bowels of programming. But, of course, nominations for the library can only come with experience which is only now maturing.

LESSONS LEARNED

The outstanding lesson for this project as a result of the MUC is that the DX system will in fact be able to meet its performance objectives in the near future, that the three-part design basis (of look-up, pattern-match, and semantically resolve) is sound.

Lesson two is: start early on the MUCs.