NEW YORK UNIVERSITY: DESCRIPTION OF THE PROTEUS SYSTEM AS USED FOR MUC-5

Ralph Grishman and John Sterling

The Proteus Project
Computer Science Department
New York University
715 Broadway, 7th Floor
New York, NY 10003
{grishman,sterling}@cs.nyu.edu

THE PROTEUS SYSTEM

History

The Proteus system which we have used for MUC-5 is largely unchanged from that used for MUC-3 and MUC-4. It has three main components: a syntactic analyzer, a semantic analyzer, and a template generator.

The Proteus syntactic analyzer was developed starting in the fall of 1984 as a common base for all the applications of the Proteus Project. Many aspects of its design reflect its heritage in the Linguistic String Parser, previously developed and still in use at New York University. The current system, including the Restriction Language compiler, the lexical analyzer, and the parser proper, comprise approximately 4500 lines of Common Lisp.

The semantic analyzer was initially developed in 1987 for the MUCK-I (RAINFORMs) application, extended for the MUCK-II (OPREPS) application, and has been incrementally revised since. It currently consists of about 3000 lines of Common Lisp (excluding the domain-specific information).

The template generator was written from scratch for the MUC-5 joint venture task; it is about 1200 lines of Common Lisp.

Stages of processing

The text goes through the five major stages of processing: lexical analysis, syntactic analysis, semantic analysis, reference resolution, and template generation (see figure 1). In addition, some restructuring of the logical form is performed both after semantic analysis and after reference resolution (only the restructuring after reference resolution is shown in figure 1). Processing is basically sequential: each sentence goes through lexical, syntactic, and semantic analysis and reference resolution; the logical form for the entire message is then fed to template generation. However, semantic (selectional) checking is performed during syntactic analysis, employing essentially the same code later used for semantic analysis.

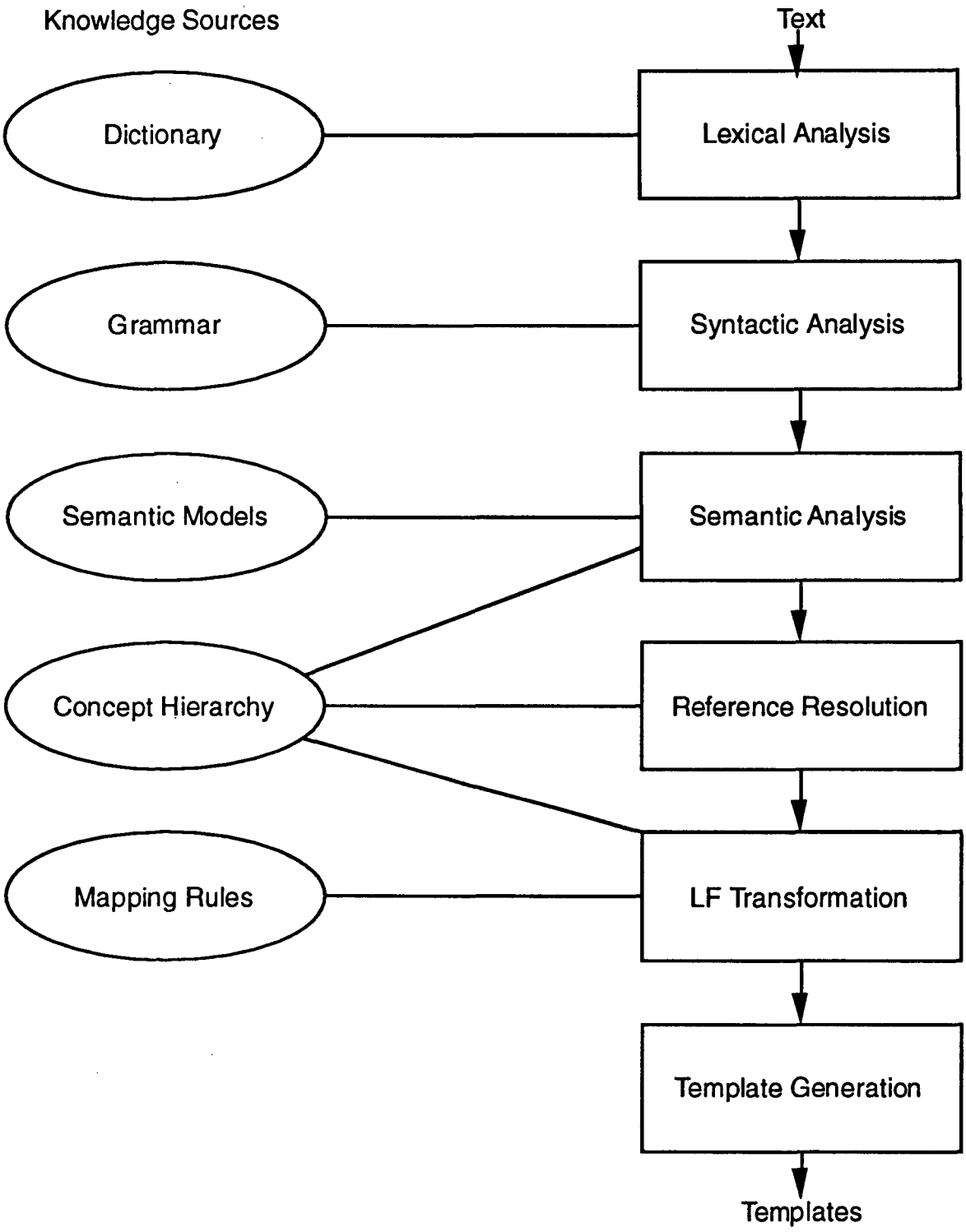


Figure 1. Proteus System Structure.

Lexical Analysis

Dictionary Format

Our dictionaries contain only syntactic information: the parts of speech for each word, information about the complement structure of verbs, distributional information (e.g., for adjectives and adverbs), etc. We follow closely the set of syntactic features established for the NYU Linguistic String Parser. This information is entered in LISP form using `noun`, `verb`, `adjective`, and `adverb` macros for the open-class words, and a `word` macro for other parts of speech:

```
(ADVERB "ABRUPTLY" :ATTRIBUTES (DSA))
(ADJECTIVE "ABRUPT")
(NOUN :ROOT "ABSCESS" :ATTRIBUTES (NCOUNT))
(VERB :ROOT "ABSCOND" :OBJLIST (NULLOBJ PN (PVAL (FROM WITH))))
```

The `noun` and `verb` macros automatically generate the regular inflectional forms.

Dictionary Files

The primary source of our dictionary information about open-class words (nouns, verbs, adjectives, and adverbs) is the machine-readable version of the Oxford Advanced Learner's Dictionary ("OALD"). We have written programs which take the SGML (Standard Generalized Markup Language) version of the dictionary, extract information on inflections, parts of speech, and verb subcategorization (including information on adverbial particles and prepositions gleaned from the examples), and generate the LISP-ified form shown above. This is supplemented by a manually-coded dictionary (about 1500 lines, 900 entries) for closed-class words, words not adequately defined in the OALD, and a few very common words. In addition, we used several specialized dictionaries for MUC-5, including a location dictionary (with all countries, continents, and major cities (CITY1 or PORT1 in the gazetteer), a dictionary of corporate designators, a dictionary of job titles, and a dictionary of currencies.

Lookup

The text reader splits the input text into tokens and then attempts to assign to each token (or sequence of tokens, in the case of an idiom) a definition (part of speech and syntactic attributes). The matching process proceeds in five steps: dictionary lookup, lexical pattern matching, spelling correction, prefix stripping, and default definition assignment. Dictionary lookup immediately retrieves definitions assigned by any of the dictionaries (including inflected forms). The specialized dictionaries are stored in memory, while the main dictionary is accessed from disk (using hashed index random access).

Lexical pattern matching is used to identify a variety of specialized patterns, such as numbers, dates, times, and possessive forms. The set of lexical patterns was substantially expanded for MUC-5 to include various forms of people's names, company names, locations, and currencies. The lexical patterns are further discussed below, in the "What's new for MUC-5" section.

If neither dictionary lookup nor lexical pattern matching is successful, spelling correction and prefix stripping are attempted. For words of any length, we identify an input token as a misspelled form of a dictionary entry if one of the two has a single instance of a letter while the other has a doubled instance of the letter (e.g., "mispelled" and "misspelled"). The prefix stripper attempts to identify the token as a combination of a prefix (e.g.,"un") and a word defined in the dictionary.

If all of these procedures fail, we assign a default definition. In mixed case text, undefined capitalized words are tagged as proper nouns; undefined lower case words are tagged as common nouns. In monocase text, all undefined words are tagged as proper nouns.

Syntactic Analysis

Syntactic analysis involves two stages of processing: parsing and syntactic regularization. At the core of the system is an active chart parser. The grammar is an augmented context-free grammar, consisting of BNF rules plus procedural restrictions which check grammatical constraints not easily captured in the BNF rules. Most restrictions are stated in PROTEUS Restriction Language (a variant of the language developed for the Linguistic String Parser) and translated into LISP; a few are coded directly in LISP [1]. For example, the count noun restriction (that singular countable nouns have a determiner) is stated as

```
WCOUNT = IN LNR AFTER NVAR:
    IF BOTH CORE Xcore IS NCOUNT AND Xcore IS SINGULAR
    THEN IN LN, TPOS IS NOT EMPTY.
```

Associated with each BNF rule is a regularization rule, which computes the regularized form of each node in the parse tree from the regularized forms of its immediate constituents. These regularization rules are based on lambda-reduction, as in GPSG. The primary function of syntactic regularization is to reduce all clauses to a standard form consisting of aspect and tense markers, the operator (verb or adjective), and syntactically marked cases. For example, the definition of `assertion`, the basic S structure in our grammar, is

```
<assertion>     ::= <sa> <subject> <sa> <verb> <sa> <object> <sa>
                        :(s !(<object> <subject> <verb> <sa*>)).
```

Here the portion after the single colon defines the regularized structure.

Coordinate conjunction is introduced by a metarule (as in GPSG), which is applied to the context-free components of the grammar prior to parsing. The regularization procedure expands any conjunction into a conjuntion of clauses or of noun phrases.

The output of the parser for the first sentence of 0592, "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." is

```
(SENTENCE
 (CENTERS
  (CENTER
   (ASSERTION
    (SUBJECT
     (NSTG
      (LNR (NVAR (NAMESTG (LNAMER (N "BRIDGESTONE" "SPORTS" "CO" "."))))))))
    (VERB (LTVR (TV "SAID")))
    (SA (SA-VAL (NSTGT (NSTG (LNR (NVAR (N "FRIDAY")))))))
    (OBJECT
     (ASSERTION (SUBJECT (NSTG (LNR (NVAR (PRO "IT"))))))
      (VERB (LTVR (TV "HAS")))
      (OBJECT
       (VENO (LVENR (VEN "SET" "up"))
        (OBJECT
         (NSTGO
          (NSTG
           (LNR (LN (TPOS (LTR (T "A")))) (NVAR (N "JOINT" "venture"))
            (RN
             (RN-VAL
              (PN (P "IN")
               (NSTGO (NSTG (LNR (NVAR (NAMESTG (LNAMER (N "TAIWAN")))))))))
             (MORE-RN
              (RN-VAL
               (PN (P "WITH")
                (NSTGO
                 (NSTG
                  (LNR
                   (LNR
                    (LN (TPOS (LTR (T "A")))
                     (APOS (APOSVAR (AVAR (ADJ "LOCAL")))))
                    (NVAR (N "CONCERN")))
                   (CONJ-WORD ("AND" "AND"))
                   (LNR
                    (LN (TPOS (LTR (T "A")))
                     (APOS (APOSVAR (AVAR (ADJ "JAPANESE")))))
                    (NVAR (N "TRADING" "house")))))))))
              (MORE-RN
               (RN-VAL
                (TOVO (TO ("TO" "TO")) (LVR (V "PRODUCE"))
                 (OBJECT
                  (NSTGO
                   (NSTG
                    (LNR (LN (NPOS (NPOSVAR (N "GOLF"))))
                     (NVAR (N "CLUBS"))))))
                 (SA
                  (SA-VAL
                   (TOVO (TO ("TO" "TO")) (LVR (V "BE"))
                    (OBJECT
                     (OBJECTBE
                      (VENPASS (LVENR (VEN "SHIPPED"))
                       (SA
                        (SA-VAL
                         (PN (P "TO")
                          (NSTGO
                           (NSTG
                            (LNR
                             (NVAR
                              (NAMESTG
                               (LNAMER
                                (N "JAPAN")))))))))))))))))))))))))
 (ENDMARK ("." ".")))
```

and the corresponding regularized structure is

```
(S SAY (VTENSE PAST)
 (SUBJECT
  (NP A-COMPANY SINGULAR (NAMES ("BRIDGESTONE" "SPORTS" "CO")) (SN NP154)))
 (OBJECT
  (S SET-UP (SUBJECT (NP IT SINGULAR (SN NP156)))
   (OBJECT
    (NP JOINT-VENTURE SINGULAR (SN NP258) (T-POS A)
     (IN (NP A-COUNTRY SINGULAR (NAMES ("Taiwan")) (SN NP163)))
     (WITH
      (AND (NP CONCERN SINGULAR (SN NP166) (T-POS A) (A-POS LOCAL))
           (NP TRADING-HOUSE SINGULAR (SN NP171) (T-POS A) (A-POS JAPANESE))))
     (RN-TOVO
      (S PRODUCE (SUBJECT ANYONE)
       (OBJECT
        (NP CLUB PLURAL (SN NP180) (N-POS (NP GOLF SINGULAR (SN NP170)))))
       (IN-ORDER-TO
        (S SHIP (SUBJECT ANYONE) (OBJECT PRO)
         (TO (NP A-COUNTRY SINGULAR (NAMES ("Japan")) (SN NP177)))))))))
   (ASPECT PERF) (VTENSE PRESENT)))
 (TIMEPREP (NP FRIDAY SINGULAR (SN NP157))))
```

The system uses a chart parser operating top-down, left-to-right. As edges are completed (i.e., as nodes of the parse tree are built), restrictions associated with those productions are invoked to assign and test features of the parse tree nodes. If a restriction fails, that edge is not added to the chart. When certain levels of the tree are complete (those producing noun phrase and clause structures), the regularization rules are invoked to compute a regularized structure for the partial parse, and selection is invoked to verify the semantic well-formedness of the structure (as noted earlier, selection uses the same "semantic analysis" code subsequently employed to translate the tree into logical form).

One unusual feature of the parser is its weighting capability. Restrictions may assign scores to nodes; the parser will perform a best-first search for the parse tree with the highest score. This scoring is used to implement various preference mechanisms:

- closest attachment of modifiers (we penalize each modifier by the number of words separating it from its head)
- preferred narrow conjoining for clauses (we penalize a conjoined clause structure by the number of words it subsumes)
- preference semantics (selection does not reject a structure, but imposes a heavy penalty if the structure does not match any lexico-semantic model, and a lesser penalty if the structure matches a model but with some operands or modifiers left over) [2,3]
- relaxation of certain syntactic constraints, such as the count noun constraint, adverb position constraints, and comma constraints
- disfavoring (penalizing) headless noun phrases and headless relatives (this is important for parsing efficiency)

The grammar is based on Harris's Linguistic String Theory and adapted from the larger Linguistic String Project (LSP) grammar developed by Naomi Sager at NYU [4]. The grammar is gradually being enlarged to cover more of the LSP grammar. The current grammar is 1600 lines of BNF and Restriction Language plus 300 lines of Lisp; it includes 186 non-terminals, 464 productions, and 132 restrictions.

Over the course of the MUCs we have added several mechanisms for recovering from sentences the grammar cannot fully parse. For MUC-5, we found that the most

effective was our "fitted parse" mechanism, which attempts to cover the sentence with noun phrases and clauses, preferring the longest noun phrases or clauses which can be identified

Semantic Analysis And Reference Resolution

The output of syntactic analysis goes through semantic analysis and reference resolution and is then added to the accumulating logical form for the message. Following both semantic analysis and reference resolution certain transformations are performed to simplify the logical form. All of this processing makes use of a concept hierarchy which captures the class/subclass/instance relations in the domain.

Semantic analysis uses a set of lexico-semantic models to map the regularized syntactic analysis into a semantic representation. Each model specifies a class of verbs, adjectives, or nouns and a set of operands; for each operand it indicates the possible syntactic case markers, the semantic class of the operand, whether or not the operand is required, and the semantic case to be assigned to the operand in the output representation. For example, the model for "<entity> forms a joint venture with <entity>" is

```
(add-clause-model :id 'clause-form
                  :parent 'clause-any
                  :constraint 'W-form-venture
                  :class 'C-form
                  :adjuncts (list (make-specifier
                                   :marker 'subject
                                   :class 'C-muc5-entity
                                   :case :agent)
                                  (make-specifier
                                   :marker 'with
                                   :class 'C-muc5-entity
                                   :case :company-list-2)
                                  (make-specifier
                                   :marker 'object
                                   :class 'C-joint-venture
                                   :essential-required 'required
                                   :relaxable nil
                                   :case :joint-venture)))
```

The models are arranged in a shallow hierarchy with inheritance, so that arguments and modifiers which are shared by a class of verbs need only be stated once. The model above inherits only from the most general clause model, clause-any, which includes general clausal modifiers such as negation, time, tense, modality, etc. The MUC-5 system has 61 clause models, 2 nominalization models, and 45 other noun phrase models, a total of about 1700 lines. The class `C-muc5-entity` in the clause model refers to the concept in the concept hierarchy, whose entries have the form:

```
(defconcept C-muc5-entity)
(defconcept C-company :typeof C-muc5-entity)
(defconcept C-government-or-country :typeof C-muc5-entity)
(defconcept C-venture :typeof C-company)
(defconcept C-joint-venture :typeof C-venture)
```

This inheritance mechanism is also used to define word classes, such as the `W-form-venture` class:

```
(defconcept W-form-venture)
(defconcept form :typeof W-form-venture)
(defconcept establish :typeof W-form-venture)
(defconcept expand :typeof W-form-venture)
(defconcept launch :typeof W-form-venture)
(defconcept set-up :typeof W-form-venture)
```

There are currently a total of 154 concepts in the hierarchy. The output of semantic analysis is a nested set of entity and event structures, with arguments labeled by keywords primarily designating semantic roles. For the first sentence of 0593, the output is

```
(EVENT
 :IDENTIFIER E001
 :TOP-LEVEL-FLAG T
 :PREDICATE C-SAY
 :TIME (ENTITY :SN NP157
               :MODEL-ID TIME-VALUE
               :IDENTIFIER N015
               :CLASS FRIDAY)
 :EVENT (EVENT
         :IDENTIFIER E003
         :TOP-LEVEL-FLAG NIL
         :PREDICATE C-FORM
         :TENSE PRESENT
         :ASPECT PERF
         :JOINT-VENTURE (ENTITY
                         :ACTIVITY (EVENT :IDENTIFIER E010
                                          :TOP-LEVEL-FLAG NIL
                                          :PREDICATE C-PRODUCE
                                          :PATIENT (ENTITY :SN NP180
                                                           :MODEL-ID NP-ANY
                                                           :SET T
                                                           :IDENTIFIER N014
                                                           :CLASS CLUB)
                                          :AGENT NIL
                                          :MODEL-ID CLAUSE-PRODUCE)
                         :AGENT (ENTITY :IDENTIFIER N009
                                        :CLASS C-MUC5-ENTITY
                                        :SET T
                                        :MEMBERS ((ENTITY :DETERMINER A
                                                          :SN NP166
                                                          :MODEL-ID NP-COMPANY-2
                                                          :IDENTIFIER N007
                                                          :CLASS CONCERN)
                                                  (ENTITY :NATIONALITY JAPANESE
                                                          :DETERMINER A
                                                          :SN NP171
                                                          :MODEL-ID NP-COMPANY-2
                                                          :IDENTIFIER N008
                                                          :CLASS TRADING-HOUSE)))
                         :NATIONALITY (ENTITY :SN NP163
                                              :NAMES ("Taiwan")
                                              :MODEL-ID NP-A-COUNTRY
                                              :IDENTIFIER N006
                                              :CLASS C-COUNTRY)
                         :DETERMINER A
                         :SN NP258
                         :MODEL-ID NP-JOINT-VENTURE-1
                         :IDENTIFIER N005
                         :CLASS C-JOINT-VENTURE)
         :AGENT (ENTITY :SN NP156
                        :MODEL-ID NP-ANY
                        :IDENTIFIER N004
                        :CLASS C-MUC5-ENTITY)
         :MODEL-ID CLAUSE-FORM)
 :AGENT (ENTITY :SN NP154
                :NAMES ("BRIDGESTONE" "SPORTS" "CO")
                :MODEL-ID NP-COMPANY
                :IDENTIFIER N0000000002
                :CLASS C-COMPANY)
 :TENSE PAST
 :MODEL-ID CLAUSE-SAY)
```

Reference resolution

Reference resolution is applied to the output of semantic analysis in order to replace anaphoric noun phrases (representing either events or entities) by appropriate antecedents. Each potential anaphor is compared to prior entities or events, looking for a suitable antecedent such that the class of the anaphor (in the concept hierarchy) is equal to or more general than that of the antecedent, the anaphor and antecedent match in number, the restrictive modifiers in the anaphor have corresponding arguments in the antecedent, and the non-restrictive modifiers (e.g., apposition) of the anaphor are not inconsistent with those of the antecedent. Special tests are provided for names, since people and companies may be referred to a subset of their full names.

Logical form transformations

The transformations which are applied after semantic analysis and after reference resolution simplify and regularize the logical form in various ways. The transformations after semantic analysis primarily standardize the attribute structure of entities so that reference resolution will work properly. The transformations after reference resolution simplify the task of template generation by casting the events in a more uniform framework and performing a limited number of inferences. For example, we show here a rule which transforms the logical form produced from "X formed a joint venture with Y" into the equivalent for "X and Y formed a joint venture":

```
(((event :predicate ?predicate
         :identifier ?id1
         :agent ?agent
         :joint-venture (entity :class C-joint-venture
                                :tied-up nil
                                :agent ?company-list-2
                                :identifier ?id2
                                . ?R2)
         . ?R1)
  (entity :identifier ?id2 . ?R3)
  (condition (isa '?predicate 'C-tie-up)))
 -->
 ((modify 1 (list :agent (conjoin-entities
                           '?agent '?company-list-2)))
  (modify 2 '(:agent nil :tied-up t))))
```

There are currently 32 such rules. These transformations are written as productions and applied using a simple data-driven production system interpreter which is part of the Proteus system.

Template generator

Once all the sentences in an article have been processed through syntactic analysis, semantic analysis, and the logical form transformations, the resulting logical forms are sent to the template generator. The logical form events and entities produced by the transformations are in close correspondence to the template objects needed for MUC-5, so the template generation is fairly straightforward. The greatest complexity was involved in the procedures for accessing the two large data bases, the gazetteer (for normalizing locations) and the Standard Industrial Classification (for classifying industries).

OUR PERFORMANCE ON MUC-5

Overall Performance

Scores

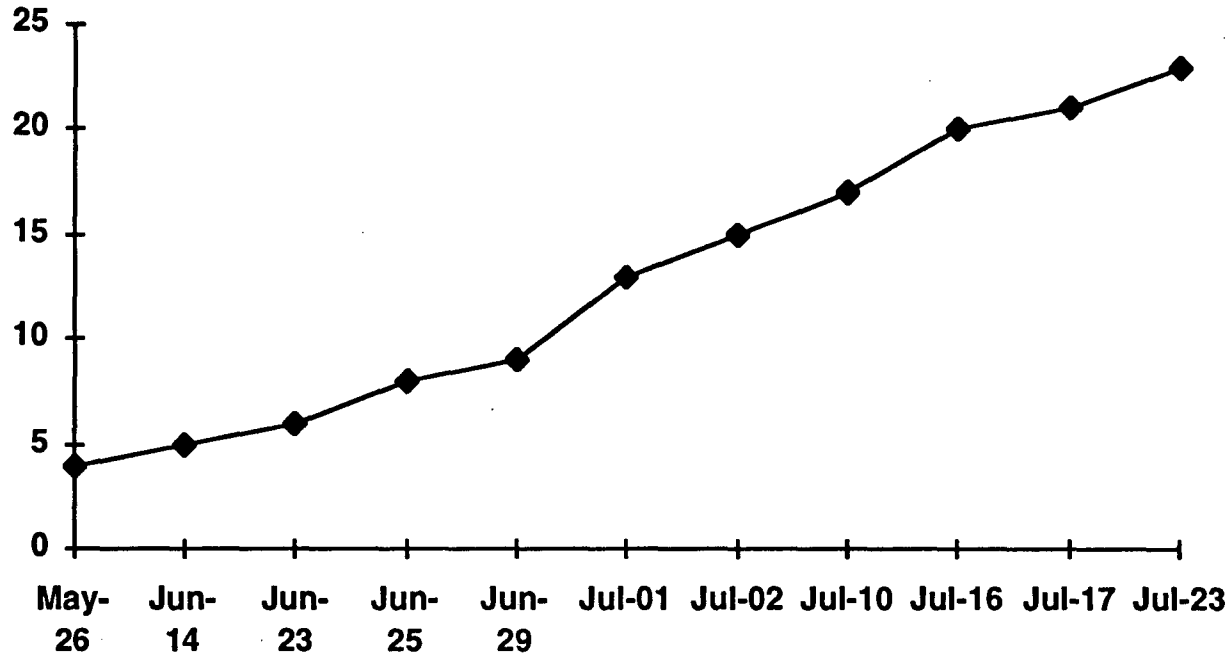
Our overall scores on the final evaluation were

Recall	22
Precision	59
F (P&R)	32
Error Rate	80

Learning Curve

Figure 2 showns how our recall gradually improved over the development period. Precision remained within a fairly narrow range, from 47 to 63, throughout the testing. Five months were available for development (March - July). One person was assigned full-time for the entire period; a second person assisted, approximately 2/3 time, for the last three months, for a total of about 7 person-months of effort (this excludes time in August preparing for the conference). March and April were devoted to getting an initial understanding of the fill rules, making minimal lexical scanner additions so that we could parse the text, developing input code to handle the different article formats, and developing some routines for larger-scale pattern matching (which were eventually not used). System integration and integrated system testing did not begin until mid-May, a couple of weeks before the dry run. Daily system testing began with a set of 25 articles, but shifted after the dry run to the first 100 dry-run messages (with the second 100 dry-run messages being used on occasion as a blind test).

Figure 2: Recall (Dry Run, part 1)



In comparison with earlier MUCs, the overhead of getting started -- understanding the fill rules, handling the different article formats, generating the more complex templates, and using the various data bases (gazetteer, SIC, currency table, corporate designator table) -- was much greater than for prior MUCs, while the manpower we had for the project was in fact somewhat less. In consequence, our system is relatively less developed than our MUC-3 system, for example. In particular, the attribute structure for the principal entity types (for MUC-5, companies) were less developed; this adversely impacted the performance of our reference resolution component and hence our event merging.

This impact was evident in our performance on the walkthrough message, 0593. We identified the primary constituent events (the joint venture and the associated ownership relations), but we failed to identify several of the co-reference relations, because of

- a bug in the handling of appositional names followed by relative clauses
- failure to do spelling correction on names (we only correct spellings to match dictionary entries)
- shortcomings in the attribute structure of company entities

Because of these problems and a weak event merging rule (compared to the more detailed rules developed for MUC-4, for example), we generated two separate tie-ups for the article, instead of one.

The system was also not tuned to any significant degree to take advantage of the MUC-5 scoring rules. Based on a suggestion by Boyan Onyshkevych, we conducted a small experiment after the conference. Because one is told in advance that almost every article in the corpus will have a reportable event, we modified the system to generate a tie-up between a Japanese company and an Indonesian company (the two most frequent nationalities in the training corpus) whenever the text analysis components were *not* able to find a tie up. This simple strategy reduced our error rate on the test corpus by 2%.

WHAT'S NEW FOR MUC-5

Lexical Analyzer

The Proteus system has a pattern matcher based on regular expressions with provision for procedural tests, which is intended for identifying lexical units before parsing. Prior to MUC-5, the system employed a small number of patterns, for structures such as dates, times, and numbers. The set of patterns was substantially enlarged for MUC-5, to include patterns for diffeent types of currencies, for company names, for people's names, for locations, and for names of indeterminate type. In mixed-case text, we used capitalization as the primary indication of the beginning of a name; in monocase text, we employed BBN's part-of-speech tagger and looked for proper noun tags.

The lexical scanner and the constraints of the lexico-semantic models acted in concert to classify names. If there was a clear lexical clue (a corporate designator at the end of a name, a title ("Mr.", "President", ...) or middle initial in a personal name), the type was assigned by the lexical scanner. If the type of a name could not be determined by the scanner, but the name occurred in a context where only one type was allowed (e.g., as the object of "own"), the type would be assigned as a side effect of applying the lexico-semantic model.

Semantic Pattern and Similarity Acquisition

We have spent considerable time over the last two years building tools to acquire semantic patterns and semantic word similarities from corpora [5, 6], and we had hoped that these would be of significant benefit in our MUC-5 efforts, particularly in broadening our system's coverage. However, we did not have much opportunity to use these tools, since so much of our time was consumed in building an initial system at some minimal performance level.

Nested Semantic Models

The lexico-sematic models as used previously specified a single level in the regularized parse tree structure: either a clause with its arguments and modifiers, or an NP with its modifiers. We have found it increasingly valuable, however, to be able to specify larger patterns which involve several parse tree levels, such as "X signed an agreement with Y to do Z", or "X formed a joint venture with Y to do Z". We have therefore extended our system in order to allow for such larger patterns, and permit the developer to specify the predicate structure into which this larger pattern should be mapped.

Model Builder

Once we began to allow these larger patterns, we found that the task of writing such patterns correctly became quite challenging. Our long-term goal is to enable a user to add such patterns, but we seemed (with the added complexity) to be moving further from this goal. We therefore implemented a "model builder" interface which allows the developer to enter a prototype sentence and the corresponding predicate structure which should be produced. The interface then creates the required lexico-semantic patterns and mapping rules.

For example, to handle constructs of the form "company signed an agreement with company to ...", the developer would enter the sentence

```
company1 signed {an agreement with company2 to act3}.
```

(where the braces, which are optional, indicate the NP bracketing) and would give the corresponding predicate

```
(c-agree :agent company1 :co-agent company2 :event act3)
```

The system would then create models and mapping rules appropriate to a sentence such as "IBM signed an agreement with Apple to form a joint venture." Since these rules apply to the syntactically analyzed sentence, they would also handle syntactic variants such as "The agreement to create the new venture was signed last week by IBM and Ford."

REFERENCES

[1] Grishman, R. PROTEUS Parser Reference Manual. PROTEUS Project Memorandum #4-C, Computer Science Department, New York University, May 1990.

[2] Grishman, R., and Sterling, J. Preference Semantics for Message Understanding. *Proc. DARPA Speech and Natural Language Workshop*, Morgan Kaufman, 1990 (proceedings of the conference at Harwich Port, MA, Oct. 15-18, 1989).

[3] Grishman, R., and Sterling, J., Information Extraction and Semantic Constraints. *Proc. 13th Int'l Conf. Computational Linguistics (COLING 90)*, Helsinki, August 20-25, 1990.

[4] Sager, N. *Natural Language Information Processing*, Addison-Wesley, 1981.

[5] Grishman, R., and Sterling, J. Acquisition of Selectional Patterns. *Proc. 14th Int'l Conf. Computational Linguistics (COLING 92),* Nantes, France, July 23-28, 1992.

[6] Grishman, R., and Sterling, J. Smoothing of Automatically Generated Selectional Constraints. *Proc. ARPA Human Language Technology Workshop,* March 21-24, 1993.

SPONSORSHIP

The development of the entire PROTEUS system has been sponsored primarily by the Advanced Research Projects Agency as part of the Strategic Computing Program, under Contract N00014-85-K-0163 and Grant N00014-90-J-1851 from the Office of Naval Research.