The NYU System for MUC-6
or
Where's the Syntax?

Ralph Grishman
Computer Science Department
New York University
grishman@cs.nyu.edu

INTRODUCTION

Over the past five MUCs, New York University has clung faithfully to the idea that information extraction should begin with a phase of full syntactic analysis, followed by a semantic analysis of the syntactic structure. Because we have a good, broad-coverage English grammar and a moderately effective method for recovering from parse failures, this approach held us in fairly good stead.

However, we were at a disadvantage with respect to groups which performed more local pattern recognition, in three regards:

1. **our systems were quite slow**

   In processing the language as a whole, our system is operating with only relatively weak semantic preferences. As a result, the process of building a global syntactic analysis involves a large and relatively unconstrained search space and is consequently quite expensive. In contrast, pattern matching systems assemble structure "bottom-up" and only in the face of compelling syntactic or semantic evidence, in a (nearly) deterministic manner.

   Speed was particularly an issue for MUC-6 because of the relatively short time frame (1 month for training). With a slow system, which can analyze only a few sentences per minute, it is possible to perform only one or at best two runs per day over the full training corpus, severely limiting debugging.

2. **global parsing considerations sometimes led to local errors**

   Our system was designed to attempt to generate a full sentence parse if at all possible. If not, it attempted a parse covering the largest substring of the sentence which it could. This global goal sometimes led to incorrect local choices of analyses; an analyzer which trusted local decisions could in many cases have done better.

3. **adding syntactic constructs needed for a new scenario was hard**

   Having a broad-coverage, linguistically-principled grammar meant that relatively few additions were needed when moving to a new scenario. However, when specialized constructs did have to be added, the task was relatively difficult, since these constructs had to be integrated into a large and quite complex grammar.

We considered carefully whether these difficulties might be readily overcome using an approach which was still based on a comprehensive syntactic grammar. It appeared plausible, although not certain, that problems (1) and (2) could be overcome within such an approach, by adopting a strategy of *conservative* parsing. A conservative parser would perform a reduction only if there was strong (usually, local) syntactic evidence or strong semantic support. In particular, chunking parsers, which built up small chunks using syntactic criteria and then assembled larger structures only if they were semantically licensed, might provide a suitable candidate. Ideally, a parser might *learn* which decisions could be safely made based purely on syntactic

evidence, but building such a parser would be a substantial research project not to be lightly undertaken in the months leading up to a MUC.

In any case, problem (3) still loomed. Our Holy Grail, like that of many groups, is to eventually get the computational linguist out of the loop in adapting an information extraction system for a new scenario. This will be difficult, however, if the scenario requires the addition of some grammatical construct, albeit minor. It would require us to organize the grammar in such a way that limited additions could be made by non-specialists without having to understand the entire grammar — again, not a simple task.

All these considerations led us to conclude that we should "do a MUC" ourselves using the pattern matching approach, in order to better appreciate its strengths and weaknesses. In particular, we carefully studied the FASTUS system of Hobbs et al. [1], who have clearly and eloquently set forth the advantages of this approach. This approach can be viewed as a form of conservative parsing, although the high-level structures which are created are not explicitly syntactic. At the end of this paper we return to the question of the relation of pattern matching to approaches which use a comprehensive grammar.

THE SYSTEM

We exaggerate, of course, the radicalness of our change since MUC-5 [4] (and since the MUC-6 dry run, which was conducted with our traditional syntactic system). Several components were direct descendants of earlier modules: the dictionary was Comlex Syntax [3]; the lexical analyzer (for names, etc.) had been gradually enhanced at least since MUC-3; the concept hierarchy code and reference resolution were essentially unchanged from earlier versions. In addition, our grammatical approach was not entirely abandoned; our noun group patterns were a direct adaptation of the corresponding portion of our grammar, just as Hobbs' patterns were an adaptation from his grammar.[1] And, as we shall see, more of the grammar crept in as our effort progressed. In essence, one could say that our MUC-6 system was built (in late August and early September, 1995) by replacing the parser and semantic interpreter of our earlier system by additional sets of finite-state patterns.

Overall Structure

The same system was used for all four MUC tasks (NE, CO, TE, and ST); the only difference lies in the information which is generated when the processing of a document is complete.

The text analysis operates in seven main stages: tokenization and dictionary look-up; four stages of pattern matching (basically, for names, noun groups, verb groups, and larger patterns); reference resolution; and output (template or SGML) generation.

Tokenization and Dictionary Look-up

Processing begins with the reading of the document and the identification of the relevant SGML-marker passages. The body of the document is divided into sentences and then into tokens. Each token is looked up in our dictionaries. For general vocabulary, we use Comlex Syntax, a broad-coverage dictionary of English developed at NYU, which provides detailed syntactic information, but does not include any proper names. This is supplemented by several specialized dictionaries, including

- a small gazetteer, which contains the names of all countries and most "major" cities
- a company dictionary, derived from the Fortune 500
- a government agency dictionary
- a dictionary of common first names
- a small dictionary of scenario-specific terms

We also use the BBN POST tagger at this stage to determine the most likely part of speech of each word.[2]

[1] And, since both these grammars can trace their origins in part to the NYU Linguistic String Grammar, the approaches here are very similar.

[2] We wish to thank BBN Systems and Technologies for providing us with this tagger.

Name Recognition

The input stage is followed by several stages of pattern matching. Each of these stages uses one or more sets of finite-state patterns to perform some reductions on the input string. The patterns are translated to LISP procedures which are then compiled, so the pattern matching can proceed very efficiently.

Each set of patterns involves one left-to-right scan of the sentence. Starting at each word, we identify the longest matching pattern (if any), use it to reduce the input sequence, and then continue with the next unmatched word.

The first set of patterns corresponds essentially to Named Entity recognition: names of people; names of companies and other organizations; locations; dates; and numeric expressions, including money and percentages.

A second, small set identifies possessive forms involving either common nouns or names as just identified.

For the MUC-6 scenario, we added a third set for names of executive positions, such as "executive director for recall and precision".

The name recognition stage records the initial mention and type of each name; subsequent mentions of a portion of that name will be recorded as aliases of the name. At this stage, a name will be recognized as being of a specific type (person, company, government organization, or other organization) if it is defined in the dictionary, if it has a distinctive form, or if it is an alias of a name of known type. (Recognition based on context is performed by subsequent stages.)

Noun Group Recognition

The second stage of pattern matching recognizes noun groups: nouns with their left modifier. In most cases, once part of speech ambiguities have been resolved (using a tagger, as we noted above), most decisions regarding noun group boundaries and structure can be made deterministically using local syntactic information. In some cases, however, the attachment can not be decided locally; in such cases, we leave the modifier unattached. For example, a present or past participle may mark the beginning of a noun group:

He enjoys driving ranges more than any golfer I know.

or may be part of a verb phrase

He enjoys driving cars.

The noun group patterns are essentially a direct transcription of that portion of our English grammar into our pattern language.

Verb Group Recognition

The third stage of pattern matching recognizes verb groups: simple tensed verbs ("sleeps"), and verbs with auxiliaries ("will sleep", "has slept", "was sleeping", etc.). Both active and passive verb forms are recognized.

Semantic Pattern Recognition

The fourth and final stage of pattern recognition involves the scenario-specific patterns. These include patterns which recognize larger noun phrase structures than simple noun groups, and patterns which recognize clausal structures.

The noun phrase patterns include noun phrase arguments, such as "president of General Motors"; apposition, such as "Fred Smith, president of General Motors"; age modifiers, such as "Fred Smith, 107 years old"; and relative clauses. Noun phrase conjunction is also handled at this stage.

The clausal patterns play the main role in this scenario, recognizing the basic events of executive succession: having jobs, starting jobs, leaving jobs, succeeding other people in jobs. Each recognized pattern is translated into an event predication in the logical form, with one of the following forms:

- start-job(person, position)

- add-job(person, position)[3]
- have-job(person, position)
- leave-job(person, position)
- leave-a-job(person) [e.g., where someone retires, but the position is not known]
- succeed(person1, person2)

For each subject-verb-object relationship, we create a separate pattern for each of the plausible syntactic forms, including the active clause, the passive clause, the relative clause (active or passive), the reduced relative clause, and the conjoined verb phrase.

These patterns also serve to resolve some type ambiguities. If we have the sentence

P. T. Barnum took the helm of F. W. Woolworth.

the system will classify "P. T. Barnum" as a person and "F. W. Woolworth" as a company.

Reference Resolution

The various stages of pattern matching produce a *logical form* for the sentence, consisting of a set of entities (for this scenario, people, organizations, and positions) and a set of events which refer to these entities. These must now be integrated with the entities and events from the prior discourse (prior sentences in the article).

Reference resolution examines each entity and event in logical form and decides whether it is an anaphoric reference to a prior entity or event, or whether it is new and must be added to the discourse representation.[4] If the noun phrase has an indefinite determiner or quantifier (e.g., "a", "some", "any", "most") it is assumed to be new information. Otherwise a search is made through the prior discourse for possible antecedents. An antecedent will be accepted if the class of the anaphor (in our classification hierarchy) is equal to or more general than that of the antecedent, if the anaphor and antecedent match in number, and if the modifiers in the anaphor have corresponding arguments in the antecedent. Special tests are provided for names, since people and companies may be referred to by a subset of their full names; a match on names takes precedence over other criteria.

Reference resolution first seeks an antecedent in the current sentence, then in the preceding sentence, then in the one before that, etc. The current sentence is scanned from right to left (i.e., the most recent antecedent is preferred). Prior sentences are scanned from left to right; this implements, in a crude way, a preference for the subjects of prior sentences.

Response Generation

For all the tasks, we use Tipster-style annotations as an intermediate representation for the information to be reported. A Tipster annotation includes a type, a set of start/end byte offsets, and a set of attributes [2]. The name recognition stage generates ENAMEX, PNAMEX, TIMEX, and NUMEX annotations as a by-product of the recognition process, so Named Entity response generation only requires that the annotations be converted to SGML. For the Coreference task, the coreference links created by reference resolution are converted to annotations and thence to SGML. For the Template Element task, the set of discourse entities is scanned for entities of the appropriate type (people and organizations), plurals and some indefinite references are eliminated, and the remainder are converted to templates.

The only substantial processing for response generation occurs in the Scenario Template task. Here a certain amount of inferencing is needed to extract the actual events from those explicitly stated in the article. For example, if the article says "Fred, the president of Cuban Cigar Corp., was appointed vice president of Microsoft." we want to infer that Fred left the Cuban Cigar Corp. This is done using the inferences

[3] *add-job* is used in situations where there is an explicit indication that the position being taken on is an additional position: "Fred was appointed to the additional post of executive vice president."

[4] In some cases, such as apposition, the anaphoric relation is determined by the syntax. Such cases are detected and marked by the pattern-matching stages, and checked by reference resolution before other tests are made.

start-job(person,job) ⇒ leave-a-job(person)

have-job(person,job) ∧ leave-a-job(person) ⇒ leave-job(person, job)

(the actual rules are a bit more complex because information about the reason for leaving a job is also required).

Response generation also handles the inferencing required for the *succeeds* predicate. If the article says "Fred was the president of Legal Beagle Inc. Fred was succeeded by Harry." we need to infer that Harry is becoming the president of Legal Beagle. If we have a predicate of the form *succeeds*(person1,person2), the system sees what other information it has about person1 or person2. If it has information about the job person2 has or is leaving, but no information about person1, it adds information about the job(s) person1 is starting. Similarly, if it has information about person1, it adds information about the job(s) person2 is leaving.

EXAMPLE

To see how these stages of processing work in concert to produce a template, consider the crucial sentences from the walkthrough article, which produce two of the three succession events:

> Mr. James, 57 years old, is stepping down as chief executive officer on July 1 and will retire as chairman at the end of the year. He will be succeeded by Mr. Dooner, 45.

The individual tokens in these sentences are gradually aggregated into larger units by the stages of processing, as follows:

dictionary look-up Dictionary look-up, combined with part-of-speech tagging, determines the syntactic features of each word. In addition, the dictionary includes some multi-word items which appear in the walkthrough sentences; these are reduced to single lexical units:

> step down
> chief executive officer

name recognition The name recognition stage identifies three units in these sentences; two names and a date:

> Mr. James
> July 1
> Mr. Dooner

In addition, name recognition records that "Mr. James" is an alias for the previously-mentioned "Robert L. James", and that "Mr. Dooner" is an alias for "John J. Dooner Jr.".

noun group recognition noun group recognition does not play a crucial role in these sentences; the only phrases which it reduces are

> the end
> the year

verb group recognition Three verb groups are recognized:

> is stepping down
> will retire
> will be succeeded

they are classified respectively as active, present tense; active, future tense; and passive, future tense verb groups.[5]

[5] Information on aspect — simple, progressive, or perfect — is not currently used by the system.

Semantic patterns: noun phrases The only noun phrase patterns — which build a noun phrase from a noun group and its modifiers — which apply to these sentences are for age modifiers:

> Mr. James, 57 years old,
> Mr. Dooner, 45

Semantic patterns: clauses Three clause-level patterns are recognized in these sentences:

> Mr. James, 57 years old, is stepping down as chief executive officer
> and will retire as chairman
> He will be succeeded by Mr. Dooner, 45

The first is an example of an active clause pattern; the second an example of a conjoined clause pattern (of the form "and"+*verb phrase*) and the third is an example of a passive pattern. Each is translated into an "event" predication in logical form (the first two with the predicate *leave-job*, the third with the predicate *succeeds*).

reference resolution Reference resolution links the "He" in the second sentence to the most recent previously mentioned person, in this case Mr. James.

response generation: inferencing at this point our discourse structure contains three predicates: Mr. James is leaving as chief executive officer; Mr. James is leaving as chairman; and Mr. Dooner is succeeding Mr. James. In processing the *succeeds* predicate, the inferencing component notes that we have explicit information on the positions that Mr. James is vacating, but not on the positions Mr. Dooner is taking on. It therefore adds event predicates asserting that Mr. Dooner is starting the jobs which Mr. James is vacating: chairman and chief executive officer.

Once this has been done, the event predicates are organized based on the company and position involved (since this is how the templates are structured), and then converted to templates.

PERFORMANCE

Overall System Performance

As one of the perpetrators of this multi-task evaluation, NYU felt obliged to participate in all four tasks. Results on the 4 tasks are as follows

Overall System Performance			
Task	Recall	Precision	F-measure (P&R)
Named Entity	86	90	88.19
Coreference	53	62	
Template Element	62	83	71.16
Scenario Template	47	70	56.40

Our relative standing on these tasks for the most part accorded with the effort we invested in the tasks over the last few months.

For Named Entity, our pattern set built on work done for previous MUCs. From mid-August to early September we spent several weeks tuning Named Entity annotation, using the Dry Run Test corpus for training, and pushed our performance to 90% recall, 94% precision on that corpus. Our results on the formal test, as could be expected, were a few points lower. There was no shortage of additional patterns to add in order to improve performance (a few are discussed in connection with our walkthrough message), but at that point our focus shifted entirely to the Scenario Template task.

For the Scenario Template task, we spent the first week studying the corpus and writing some of the basic code needed for the pattern-matching approach, which we were trying for the first time. The remainder of the time was a steady effort of studying texts, adding patterns, and reviewing outputs. Our first run was made 10 days into the test period; we reached 29% recall one week after the first run and 48% two weeks after the

first run; our final run on the training corpus reached 54% recall (curiously, precision hovered close to 70% throughout the development period).

For the final system, we attempted to fill all the slots, but did not address some of the finer details of the task. We did not record "interim" occupants of positions, did not do the time analysis required for ON_THE_JOB (we just used NEW_STATUS), and did not distinguish related from entirely different organizations in the REL_OTHER_ORG slot. In general, it seemed to us that — given the limited time — adding more patterns yielded greater benefits than focusing on these details.

NYU did relatively well on the Scenario Template task. We can hardly claim that this was the result of a new and innovative system design, since our goal was to gain experience and insight with a design which others had proven successful. Perhaps it was a result of including patterns beyond those found in the formal training. In particular, we

- added syntactic variants (relatives, reduced relatives, passives, etc.) of patterns even if the variants were not themselves observed in the training corpus
- studied some 1987 Wall Street Journal articles related to promotions (in particular, we searched for the phrase "as president"), and added the constructs found there

Perhaps we just stayed up late a few more nights than other sites.

We did not do any work specifically for the Coreference and Template Element tasks, although our performance on both these tasks gradually improved as a result of work focussed on Scenario Templates.

Performance on the Walkthrough Message

Performance on the walkthrough message was not very different from that on the test corpus as a whole:

Performance on the Walkthrough Message		
Task	Recall	Precision
Named Entity	81	90
Coreference	68	78
Template Element	54	79
Scenario Template	60	56

Examining the output from Named Entity showed a variety of errors, including

- interpreting "60 pounds" as money rather than weight
- not having clubs as organizations, so we didn't peg "New York Yacht Club" as an organization
- not knowing that "Coke" can be another name for "Coca-Cola" (a real problem, since "Coke" occurs several times in the text)
- not having "Hollywood" in our gazetteer
- errors with names including apostrophes, including "Taster's Choice" and "I Can't Believe It's Not Butter"

In addition, the walkthrough article pointed out a bug in the propagation of type information from initial mentions to subsequent mentions of a name.

Two errors accounted for most of our incorrect slots on the Scenario Template task. First, we did not have "hire" among our set of *appoint* verbs, which included "appoint", "name", "promote", and "elect"; this caused us to lose one entire succession event. (It also led to NE and TE errors, since we didn't have the context pattern "hired from ..." which would have led us to tag "J. Walter Thompson" as a company in the phrase "hired from J. Walter Thompson".) Second, we generated duplicate (spurious) instances of the IN_AND_OUT templates for the "chief executive officer" position.

THE ROLE OF SYNTAX

The goal we had set for ourselves was to "do a MUC" using the pattern-matching approach, in order to better understand the relative strengths and weaknesses of the pattern-matching (partial parsing) and the full-parsing approaches. We consider ourselves successful in meeting this goal; we implemented the pattern-matching scheme quickly and did quite well in generating Scenario Templates. And the approach did indeed mitigate the shortcomings of the full parsing approach which we outlined in the introduction.

We also experienced first-hand some of the shortcomings of the partial-parsing, semantic pattern approach. Syntax analysis provides two main benefits: it provides generalizations of linguistic structure across different semantic relations (for example, that the structure of a main clause is basically the same whether the verb is "to succeed" or "to fire"), and it captures paraphrastic relations between different syntactic structures (for example, between "X succeeds Y", "Y was succeeded by X", and "Y, who succeeded X"). These benefits are lost when we encode individual semantic structures. In particular, in our system, we had to separately encode the active, passive, relative, reduced relative, etc. patterns for each semantic structure. These issues are hardly new; they have been well known at least since the syntactic grammar vs. semantic grammar controversies of the 1970's.

How, then, to gain the benefits of clause-level syntax within the context of a partial parsing system? One approach, which we have implemented in the weeks since MUC, has been clause level patterns which are expanded by *metarules.*[6]

As a simple example of a clause-level pattern, consider

```
(defclausepattern runs
    "np-sem(C-person) vg(C-run) np-sem(C-company):
     person-at=1.attributes, verb-at=2.attributes,
     company-at=3.attributes"
    when-run)
```

This specifies a clause with a subject of class *C-person*, a verb of class *C-run* (which includes "run" and "head"), and an object of class *C-company.*[7] This is expanded into patterns for the active clause ("Fred runs IBM"), the passive clause ("IBM is run by Fred."), relative clauses ("Fred, who runs IBM, ..." and "IBM, which is headed by Fred, ..."), reduced relative clauses ("IBM, headed by Fred, ...") and conjoined verb phrases ("... and runs IBM", "and is run by Fred").

Using `defclausepattern` reduced the number of patterns required and, at the same time, slightly improved coverage because — when we had been expanding patterns by hand — we had not included all expansions in all cases.

The use of clause-level syntax to generate syntactic variants of a semantic pattern is even more important if we look ahead to the time when such patterns will be entered by users rather than computational linguists. We can expect a computational linguist to consider all syntactic variants, although it may be a small burden; we cannot expect the same of a typical user.

We expect that users would enter patterns by example, and would answer queries to create variants of the initial pattern. As a first step in this direction, we have coded a non-interactive `pattern-by-example` procedure which takes a sentence which is prepared as an exemplar of a pattern, analyzes it with the stages of pattern matching described above, and then converts the resulting units to elements of a pattern. For example.

```
(pattern-by-example
  :exemplars ("Fred Smith became president."
              "Fred Smith began as president."
              "Fred Smith assumed the position of president."
```

[6] These have some kinship to the metarules of GPSG, which expand a small set of productions into a larger set involving the different clause-level structures.

[7] It also specifies that the attributes of these three constituents are to be bound to the variables `person-at`, `verb-at`, and `company-at`, and that the procedure `when-run` is to be invoked when this pattern is matched.

```
            "Fred Smith took over as president.")
 :event (:predicate start-job
         :person "Fred"
         :position "president")
 :type-assertions ((person "Fred"))
 :no-pass t)
```

Each of the four exemplars would be converted to a pattern; the system would recognize that this has the basic form of a clause and generate a corresponding **defclausepattern** which generates the predicate given by **:event**. In order for this to be a viable entry procedure for non-specialists, this will have to be made into an interactive interface and difficult issues will have to be addressed about how sentence constituents should be appropriately generalized to create pattern elements. However, this begins to suggest how users without detailed system knowledge might be able to create suitable patterns.

This most recent explorations also indicate how syntax can "creep back" into a system from which it was unceremoniously ejected. In the pattern matching approach, we no longer have a monolithic grammar, but we are now able to take advantage of the syntactic regularities of both noun phrases and clauses. Noun group syntax remains explicit, as one phase of pattern matching. Clause syntax is now utilized in the metarules for defining patterns and in the rules which analyze example sentences to produce patterns.

ACKNOWLEDGEMENTS

I want to thank John Sterling for his assistance with many of the details of the MUC evaluation, and allowing me to survive yet another MUC.

The development of our language analysis software and our participation in the MUCs has been supported by the Advanced Research Projects Agency under a series of contracts. This work is currently supported under Contract 94-FI57900-000 from the Office of Research and Development and under Contract DABT63-93-C-0058 from the Department of the Army.

References

[1] D. Appelt, J. Hobbs, J. Bear, D. Israel, M. Kameyama, and M. Tyson. SRI: Description of the JV-FASTUS System used for MUC-5. In *Proc. Fifth Message Understanding Conf. (MUC-5)*, Baltimore, MD, August 1993. Morgan Kaufmann.

[2] Ralph Grishman. Tipster Phase II Architecture Design Document, Version 1.52. New York University, August 1995.

[3] Ralph Grishman, Catherine Macleod, and Adam Meyers. Comlex Syntax: Building a computational lexicon. In *Proc. 15th Int'l Conf. Computational Linguistics (COLING 94)*, pages 268–272, Kyoto, Japan, August 1994.

[4] Ralph Grishman and John Sterling. New York University: Description of the PROTEUS System as used for MUC-5. In *Proc. Fifth Message Understanding Conf. (MUC-5)*, Baltimore, MD, August 1993. Morgan Kaufmann.