Description of the LINK System Used for MUC-5

Steven L. Lytinen, Robert R. Burridge,
Peter M. Hastings, and *Christian Huyck*
Artificial Intelligence Laboratory
The University of Michigan
Ann Arbor, MI 48109
E-mail: lytinen@engin.umich.edu

BACKGROUND

Over the past five years, we have developed a natural language processing (NLP) system called LINK. LINK is a unification-based system, in which all syntactic and semantic analysis is performed in a single step. Syntactic and semantic information are both represented in the grammar in a uniform manner, similar to HPSG (Pollard and Sag, 1987).

LINK has been used in several information extraction applications. In a project with General Motors, LINK was used to process terse free-form descriptions of symptoms displayed by malfunctioning automobiles, and the repairs which fixed them. In this very narrow domain, LINK achieved recall and precision rates of 80-85%.

Most recently, we used the LINK system to participate in MUC-4. During this competition, we developed initial versions of pre and postprocessing modules which were further developed in MUC-5. In MUC-4, LINK achieved recall and precision rates of about 40%.

FLOW OF CONTROL

In the spectrum of information extraction approaches represented in MUC-5, LINK tends toward computing a complete syntactic and semantic analysis of each sentence. The main module of the system is a unification-based chart parser. Relatively little preprocessing is performed on individual sentences before they are passed to the parser. A complete analysis of each sentence is attempted, although partial parses are utilized if a complete parse cannot be produced.

The overall system consists of the modules shown in figure 1. One sentence at a time passes through the modules in the order shown in the figure. Each module's function is described below.

The Tokenizer

The tokenizer produces LISP-readable files from a 100-article source file. Each file consists of header information followed by the sentences of the article represented as lists of tokens. Tokens that have special meaning in LISP, such as the single and double quotes, commas, and periods are modified to be readable by the main parsing engine.

Sentence boundaries are hypothesized whenever a period is seen. An exception to this is if a period follows a known abbreviation, and is not followed by a capitalized token, then it is not the end of the sentence.

Double quotes are simply removed, and single quotes that are used as quotation markers are removed. Contractions are expanded and possessives are made into separate tokens (e.g.,

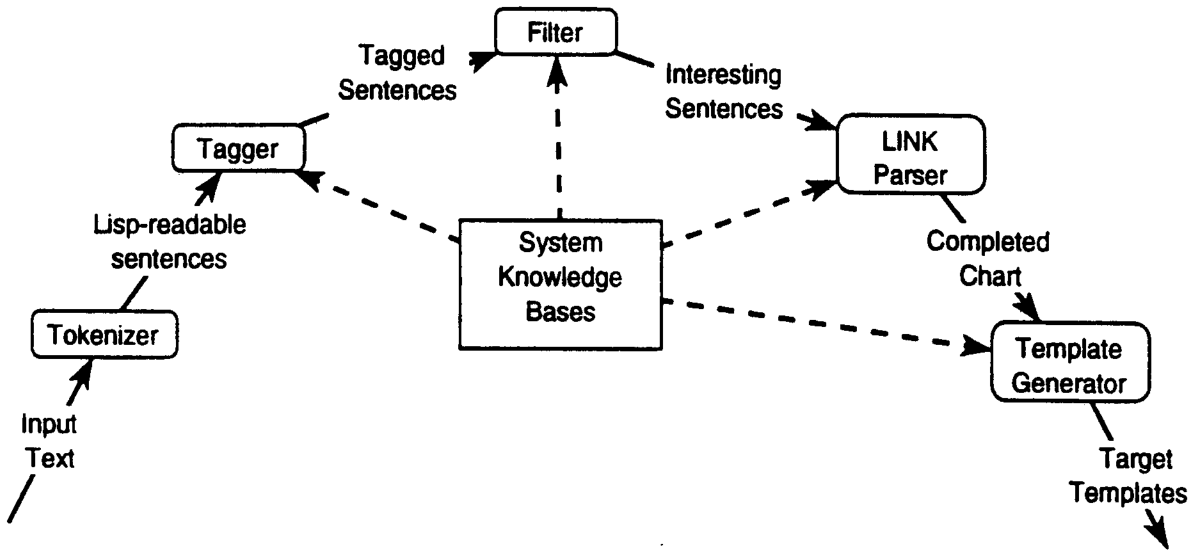

Figure 1: Modules of the MUC-5 LINK system

"Nikon's" → "Nikon *'S*"). Other special LISP symbols are converted to LISP-readable symbols.

The Tokenizer checks the case of each word, and puts sequences of capitalized words inside strings for the use of the Tagger, as described below. It also breaks apart hyphenated tokens if the first half is a number (e.g., "25-Mhz"), to allow the grammar access to the units.

The Tokenizer also performs some filtering tasks. Names of locations at the beginning of the text or abstract are removed, as are author name lines. and COMLINE tag lines. Sentences that are too short to be interesting are removed.

The Tagger

Because the input is mixed case in this domain, and because many of the proper names that would normally be unknown to the system lexicon are capitalized, the MUC-5 LINK system uses a pre-parse tagger to process and attempt to identify capitalized words which are passed as strings from the Tokenizer. The Tagger uses heuristics (aka hacks) to break apart strings in several different ways. Some of the tags that are used include: :COMP-NAME for things that seem to be obviously company names, :LOCATION for city/state pairs, :PERSON-NAME for people names (if they have Mr, Mrs, VP, Dr in front), and :NAME for other names.

Some example rules that the tagger uses are:

1. If a word is a known acronym (e.g. DRAM) or an abbreviation that is normally capitalized (e.g. "Mbit"), then just pass the word as a regular lexeme.

2. If the string ends in a word like "Corp" or "Co", tag the string as a company name.

3. If a string is followed by a word like "President" or "Spokesman" and then another string, make the first part a company name and the rest a person name.
4. If a string is followed by a comma and then a state name, tag it as a city / state pair

The Filter

Our filtering mechanism allows the system to ignore all sentences which have no useful meaning. Each sentence in an article is checked to see if it contains at least one word whose meaning is relevant to the domain; if so, the sentence is passed on to the parsre. Words with relevant meanings to this domain included verbs indicating the development or purchase of a microelectronics capability (e.g., "transfer" or "use"); names of companies or people; and various nouns of interest (e.g., "device", "hydrofluoric", "temperature" and "DRAM").

The LINK parser

LINK is unification-based chart parser, which parses a sentence at a time. The LINK parser applies unification grammar rules to a sentence to generate a syntactic and semantic representation. A set of principled grammar rule application heuristics select which grammar rules to apply. If these heuristics fail, we revert to bottom-up chart parsing. We will outline the format of the grammar and then we will describe our parsing strategy.

The LINK grammar

LINK's grammar rules are quite similar in form to those used in PATR-II (Shieber, 1986). Semantic information resides mainly in the lexicon, along the lines of HPSG (Pollard and Sag, 1987). This organization improves the portability of the system, since the vast majority of the grammar should be applicable to other domains, while the lexicon contains most of the domain-specific information.

The integration of syntactic and semantic knowledge into the same grammar formalism is crucial to our system's ability to process large texts in a reasonable length of time, and to producing the semantic analysis used to generate templates.

Edges are placed in the chart to represent constituents that the parser identifies. Edges have associated with them both syntactic and semantic information, represented in the form of a directed acyclic graph (DAG). The DAGs correspond to the information in the set of grammar rules used to build a constituent.

The MUC-5-LINK parsing strategy

LINK is a bottom-up chart parser which does not use top-down constraints. Top-down constraints are not used so that as many partial parses as possible can be generated.

Because unrestricted bottom-up chart parsing can be (and is, in our system) very inefficient, LINK uses heuristics to decide on the next edge to be entered into the chart. Many of the heuristics we use are taken from those suggested in psycholinguistic work (e.g., Ford, Bresnan, and Kaplan, 1982), although we found the need to embellish these with additional heuristics of our own (see Huyck and Lytinen, 1993, for details).

The heuristics are encoded in a rule-based system. The rules are invoked each time a new edge is to be entered into the chart, in case more than one edge could be entered next. Each rule specifies a set of conditions under which a grammar rule should be preferred or unpreferred. Rules may specify several different types of preference levels, similar to the preferences that are used in SOAR (Laird, Rosenbloom, and Newell, 1987). Heuristics may state that one grammar rule is preferable to another under some set of circumstances (i.e., if it is possible to apply both rule a and rule b at this point, then rule a should be applied), that a rule is a good candidate, that it is a bad candidate, or that it is the best candidate (i.e., under these conditions, definitely apply this grammar rule).

Because the heuristics are incomplete, often it is the case that, at some point during the parse, they are not able to suggest which rule to apply next. When this occurs, the system performs regular undirected bottom-up parsing. This continues until a complete parse of the sentence is found, no more rules can be applied, or a maximum time limit is exceeded. If a complete parse is not found, one or more partial parses is passed on for further processing. No attempt is made to "patch" together a complete interpretation of the sentence if it is not parsed successfully.

The Postprocessor

The postprocessor is responsible for assembling the semantic representations of individual sentences into a coherent representation of the entire article, and for generating the response template(s) from this overall representation. Our MUC-5 postprocessor is a two-stage, rule-based system. In the first stage, the rules transform representations produced by the parser into a cannonical form. Irrelevant portions of the representation are also discarded in this first stage. In the second stage, another set of rules transforms these representations into a form which much more closely resembles the form of the response templates.

A rule consists of a left hand side (lhs), which must match (i.e., unify with) the semantic output from the parser. If the lhs matches, the representation is converted to the form specified in the right hand side (rhs).

Here are some example rules from the first stage of postprocessing:

```
(CONVERT report-action
        (lhs) = report
        (lhs object) = ACTION
        (rhs) = (lhs object)
        (rhs actor) = (lhs actor))

(CONVERT equiv
        (lhs) = equiv
        (lhs actor) = HUMAN
        (lhs actor name) = customer
        (rhs) = transfer-to-customer
        (rhs recipient) = (lhs object))
```

The first rule converts the representation produced for sentences such as "It was reported that ...". If the main predicate representing the sentence is REPORT, and the reported object is an ACTION, then this rule discards the REPORT predicate and replaces it with the ACTION. If the ACTION has no actor, it is filled in as the actor of the REPORT. Thus, the transformed representation of the sentence "LSI Logic Corp. reported that they developed..." becomes DEVELOP, with the actor filled in as "LSI Logic Corp."

The second rule transforms the representation of a sentence such as "The customer was Hampshire Instruments." Whenever the main predicate is EQUIV (our semantic representation of "to be"), and the subject (or actor) of this action is "customer", this rule converts the representation to the predicate TRANSFER-TO-CUSTOMER, the recipient of which is the complement (object) of "to be". Together, these two rules transform the representation of a sentence like "The customer is reported to be LSI Logic Corp" to the predicate TRANSFER-TO-CUSTOMER, the recipient of which is "LSI Logic Corp."

The postprocessor also merges representations from separate sentences into a single template when appropriate. After the transformation rules are run, the representations of two sentences are merged together if they can unify. The resulting single representation is simply the result of the unification. If representations of sentences cannot be unified, then their representations may produce separate templates in the response.

SYSTEM WALKTHROUGH

We now describe our system's processing of the walkthrough article, 2789568:

> In the second quarter of 1991, Nikon Corp. (7731) plans to market the "NSR-1755EX8A," a new stepper intended for use in the production of 64- Mbit DRAMs. The stepper will use an 248-nm excimer laser as a light source and will have a resolution of 0.45 micron, compared to the 0.5 micron of the company's latest stepper. Nikon will price the excimer laser stepper at 300-350 million yen, and the company expects to sell 50 systems during the initial year of marketing.

The response generated by LINK for this article and the answer key are shown in figures 2 and 3.

We will describe the behavior of each module on the example article. The tokenized walkthrough file is shown below:

```
(who-templated ())
(document-no (2789568))
(date (October 19 |,| 1990 ))
(reported-by ("Comline Electronics "))
```

```
<TEMPLATE-2789568-1> :=
   DOC NR: 2789568
   DOC DATE: 191090
   DOCUMENT SOURCE: "Comline Electronics"
   CONTENT: <MICROELECTRONICS_CAPABILITY-2789568-31705>
   EXTRACTION TIME: 0
   DATE TEMPLATE COMPLETED: 230893
<MICROELECTRONICS_CAPABILITY-2789568-31705> :=
   PROCESS: <LITHOGRAPHY-2789568-31706>
<LITHOGRAPHY-2789568-31706> :=
   TYPE: LASER
   DEVICE: <DEVICE-2789568-31696>
   EQUIPMENT: <EQUIPMENT-2789568-31697>
<EQUIPMENT-2789568-31697> :=
   EQUIPMENT_TYPE: STEPPER
<DEVICE-2789568-31696> :=
   FUNCTION: DRAM
   SIZE: (64 MBITS)
```

Figure 2: LINK's response for article 2789568

```
(In the second quarter of 1991 |,| "Nikon Corp" |(| 7731 |)| plans to market
 the "NSR-1755EX8A" |,| a new stepper intended for use in the production
 of 64 "Mbit DRAMs" |.|)

(The stepper will use an 248 nm excimer laser as a light source and will have
 a resolution of 0.45 micron |,| compared to the 0.5 micron of the company
 |'S| latest stepper |.|)

(Nikon will price the excimer laser stepper at 300 to 350 million yen |,| and
 the company expects to sell 50 systems during the initial year of marketing
 |.|)
```

All three of the sentences from the walkthrough example are passed through the filter for further processing. The first sentence mentions "Nikon Corp" and has other meaningful words; the second sentence has the word "use" and other meaningful words; and the third sentence has the word "company" along with other meaningful words.

Quoted strings are further analyzed by the tagger, to determine what type of object they are likely to be. The completely tagged walkthrough file is shown below:

```
(IN THE SECOND QUARTER OF 1991 |,| (:COMP-NAME NIKON CORP) |(| 7731 |)| PLANS
 TO MARKET THE (:NAME NSR-1755EX8A) |,| A NEW STEPPER INTENDED FOR USE IN THE
 PRODUCTION OF 64 MBIT DRAMS \.)
```

```
<TEMPLATE-2789568-1> :=
    DOC NR: 2789568
    DOC DATE: 191090
    DOCUMENT SOURCE: "Comline Electronics"
    CONTENT: <MICROELECTRONICS_CAPABILITY-2789568-1>
             <MICROELECTRONICS_CAPABILITY-2789568-2>
    DATE TEMPLATE COMPLETED: 031292
    EXTRACTION TIME: 7
    COMMENT: / "TOOL_VERSION: LOCKE.5.2.0"
           / "FILLRULES_VERSION: EME.5.2.1"
<MICROELECTRONICS_CAPABILITY-2789568-1> :=
    PROCESS: <LITHOGRAPHY-2789568-1>
    MANUFACTURER: <ENTITY-2789568-1>
<MICROELECTRONICS_CAPABILITY-2789568-2> :=
    PROCESS: <LITHOGRAPHY-2789568-2>
    MANUFACTURER: <ENTITY-2789568-1>
    DISTRIBUTOR: <ENTITY-2789568-1>
<ENTITY-2789568-1> :=
    NAME: Nikon CORP
    TYPE: COMPANY
<LITHOGRAPHY-2789568-1> :=
    TYPE: LASER
    GRANULARITY: ( RESOLUTION 0.45 MI )
    DEVICE: <DEVICE-2789568-1>
    EQUIPMENT: <EQUIPMENT-2789568-1>
<LITHOGRAPHY-2789568-2> :=
    TYPE: UNKNOWN
    GRANULARITY: ( RESOLUTION 0.5 MI )
    EQUIPMENT: <EQUIPMENT-2789568-2>
<DEVICE-2789568-1> :=
    FUNCTION: DRAM
    SIZE: ( 64 MBITS )
<EQUIPMENT-2789568-1> :=
    NAME_OR_MODEL: "NSR-1755EX8A"
    MANUFACTURER: <ENTITY-2789568-1>
    MODULES: <EQUIPMENT-2789568-3>
    EQUIPMENT_TYPE: STEPPER
    STATUS: IN_USE
<EQUIPMENT-2789568-2> :=
    MANUFACTURER: <ENTITY-2789568-1>
    EQUIPMENT_TYPE: STEPPER
    STATUS: IN_USE
<EQUIPMENT-2789568-3> :=
    MANUFACTURER: <ENTITY-2789568-1>
    EQUIPMENT_TYPE: RADIATION_SOURCE
    STATUS: IN_USE
```

Figure 3: Answer key for article 2789568

```
(THE STEPPER WILL USE AN 248 NM EXCIMER LASER AS A LIGHT SOURCE AND WILL HAVE
 A RESOLUTION OF 0.45 MICRON |,| COMPARED TO THE 0.5 MICRON OF THE COMPANY
 |'S| LATEST STEPPER \.)

(NIKON WILL PRICE THE EXCIMER LASER STEPPER AT 300 TO 350 MILLION YEN |,| AND
 THE COMPANY EXPECTS TO SELL 50 SYSTEMS DURING THE INITIAL YEAR OF MARKETING
 \.)
```

The tagger has used the company indicator "Corp" to specify "Nikon Corp" as a company name. "NSR-1755EX8A" was not in the lexicon, nor did it have any additional indicators, so it was assumed (correctly) to be a proper name. The string "Mbit DRAMs" was not tagged because each word is known to the tagger to be an acronym / abbreviation. These words are simply passed along, and the lexicon provides the appropriate information for them.

Before parsing, the chart for the parser (as described below) is built adding constituents for each word or tagged item. When the parser reads a tagged item from the input sentence, it simply makes an entry in the chart at that position with the semantic type corresponding to the tag and the words contained in the item. For example, (:COMP-NAME NIKON CORP) turns into an entry with type Company, and name "Nikon Corp".

The parser is not successful at completely parsing any of these sentences. This primarily because the grammar and lexicon are lacking several necessary pieces of information. In the first sentence, "plan" is not marked in the lexicon as taking an infinitival complement. Thus, the construction cannot be parsed. There is also no grammar rule for parsing a determiner followed by a name as a noun phrase ("the NSF-1766EX8A"). Had this sentence read, "... market the NSF-1766EX8A stepper," the partial parse would have been more complete. As it is, only the following information can be extracted from this sentence:

```
ENTITY
   NAME : Nikon Corp
   TYPE : COMPANY

NAME
   NAME : Nsr-1755ex8a

DEVICE
   FUNCTION : DRAM
   SIZE : LENGTH
      NUM : PLURAL
         VALUE : *64*
      SCALE : MBITS

EQUIPMENT
   EQUIPMENT_TYPE : STEPPER
```

Except for "market", none of the verbs in this sentence were defined in our lexicon as interesting; thus, none of them are included in the partial parses sent on to the postprocessor.

Because "Nikon Corp" and the name of the stepper are not attached to anything, the postprocessor does not know where in the final template these should be placed. Thus, they are discarded. STEPPER, however, results in the production of a LITHOGRAPHY template, and the DRAM is attached as the DEVICE, resulting in the response shown in figure 2.

No additional information is extracted from sentences 2 and 3. In sentence 2, the text "will have a resolution of 0.45 micron, compared to the 0.5 micron of the company's latest stepper" was not parsed well enough for the system to realize that 2 different steppers are being described. Granularity specifications were not handled well by the postprocessing rules. Had the granularities been successfully attached to the representations of the two steppers, then our system would have produced two different LITHOGRAPHY templates, because different granularities would have caused unification of the two steppers to fail. Thus, the response would have contained two separate templates. However, the granularities were not successfully incorporated into the templates, resulting the steppers being merged into a single template.

The final sentence provides another opportunity to identify "Nikon Corp" as being the MANUFACTURER and DISTRIBUTOR of the LITHOGRAPHY technique. However, again, the word "price" was not defined in our lexicon as a verb relevant to the domain, so the information was ignored.

ANALYSIS OF PERFORMANCE

The LINK system's performance on the MUC-5 English microelectronics test set is shown in figure 4. Our system's performance is relatively precision-oriented. We suspect that this is due to the fact that our approach attempts complete analyses of each sentence. Thus, information which is extracted is relatively reliable, while additional information may be missed.

	Rec	Pre	Und	Ovg
ALL OBJECTS	16	39	76	41
MATCHED ONLY	43	63	44	19
TEXT FILTERING	99	75	1	25

	P&R	2P&R	P&2R
F-MEASURES	22.75	30.27	18.22

	ERR	UND	OVG	SUB
ALL OBJECTS	86	76	41	34
MATCHED ONLY	62	44	19	22

Figure 4: Performance of LINK on the MUC-5 English Microelectronics test set

Our system was tunable its use of partial parses that were used to generate templates. In its most conservative setting, only partial parses whose semantic interpretations involved important actions (e.g., DEVELOP, SELL, etc.) were used in postprocessing. The system could be made less conservative by expanding the types of partial parses that were used in tempalte generation. In its least conservative setting, even single words might be chosen as interesting partial parses, resulting in the generation of a template. For example, the appearance of the word "CVD" could result in the generation of a LAYERING template with TYPE field CVD.

For the test run, we used the system in its least conservative setting. During development testing, we found that this setting resulted in approximately 50% improvement in recall rates without adversely affecting precision. We believe that this reflects the English microelectronics domain. Since the vocabulary used in articles in this domain consisted of a large number of technical terms not normally used in most English texts, the extraction of information based on occurrence of these words without analysis of their surrounding context was a relatively safe thing to do. In other domains, it is likely that the use of single-word partial parses would result in significant reduction in precision.

Our system's precision results did suffer from the fact that templates were sometimes produced that contained so little information that they could not be matched by the scorer to answer key templates. These templates were counted by the scorer as spurious, reducing our precision score. We plan to analyze our results further to calculate the system's precision had it not produced these unmatchable templates.

Of interest is our system's performance on text filtration. The 99% recall, 75% precision performance is much higher than what might be expected given LINK's overall recall/precision rates. We suspect that these results are due to our system's full-analysis approach.

Our system is far from mature. Due to lack of resources this year, the total development time for the system totaled only about 6 person-months. This represents about 1/3 of the development time of our MUC-4 system. Thus, the knowledge base of the system is still quite incomplete. This resulted in the low recall performance of the system. Further development of the knowledge base is likely to greatly improve system performance.

System Training

We used two specialized techniques to aid in the development of the system knowledge bases. The first was to use the development keys as a sort of pocket dictionary for some of the important and often-used words. We did this by extracting all the slot fillers and their types from the templates. For all the string fills, we added the string directly to the dictionary with the semantic type that was derived from the slot that it filled. Many of the set fills were also added verbatim to the lexicon, since in this domain set fills were often technical terms (e.g., CVD). Other lexicon entries were simply created by either expanding the set-fill abbreviations or abbreviating the full-text set-fills.

The other main training source came as a result of the tagger. Since the tagger made it possible to recognize proper names that were not in the lexicon by analyzing strings of capitalized words, we used the tagged items to hypothesize new lexicon entries. This was only done for items that the tagger was sure of, like company names (strings that ended with "Corp", "Co", "Inc", etc) and person names that started with "Mrs", "Dr", "VP", etc. These definitions were not entered directly into the lexicon, but were put into a separate file so that they could be reviewed by a knowledge engineer.

CONCLUSION

Although LINK's performance on the English Microelectronics testset was less then stellar, it is difficult to draw conclusions about the use of our approach on this domain. The primary reason for degradation of performance as compared to MUC-4 was a lack of resources needed to develop a proper knowledge base for the domain. Lack of information in the lexicon, grammar, and system's domain knowledge resulted in poor analysis of the majority of articles.

The system's relatively good performance in precision indicates that our full-analysis approach is likely to yield reliable results when information is extracted. However, the Microelectronics domain may be unusual in the frequency of technical terms which are not commonly used in general English. Because of this property of the domain, it appears that techniques relying on less complete analysis of the text may be appropriate also. Our own experience indicated that the utilization of even partial parses of only a few words (or even a single word) improved our system's recall without damaging precision. We believe that in a domain with a less specialized vocabulary, techniques relying on specific keywords would be more likely to degrade precision performance.

REFERENCES

Ford, M., Bresnan, J., and Kaplan. R. (1982). A competence-based theory of syntactic closure. In Bresnan, J. (ed), *The Mental Representation of Grammatical Relations.* Cambridge, MA: MIT Press.

Huyck, C., and Lytinen, S. (1993). Efficient heuristic natural language parsing. In *Proceedings of the Eleventh National Conference on Artificial Intelligence,* Washington, D.C., July 1993.

Shieber, S. (1986). *An Introduction to Unification-Based Approaches to Grammar.* Stanford, CA: Center for the Study of Language and Information.

Pollard, C., and Sag, I. (1987). *Information-based Syntax and Semantics.* Stanford, CA: Center for the Study of Language and Information.