USC: DESCRIPTION OF THE SNAP SYSTEM USED FOR MUC-5

Dan Moldovan, Seungho Cha, Minhwa Chung, Tony Gallippi, Kenneth J. Hendrickson, Jun-Tae Kim, Changhwa Lin, and Chinyew Lin

Parallel Knowledge Processing Laboratory
Department of EE-Systems
University of Southern California
Los Angeles, California 90089-2562

INTRODUCTION

Background

The SNAP information extraction system has been developed as a part of a three-year SNAP project sponsored by the National Science Foundation. The main goal of the SNAP project is to build a massively parallel computer capable of fast and accurate natural language processing [5]. Throughout the project, a parallel computer was built in the Parallel Knowledge Processing Laboratory at USC, and various software was developed to operate the machine [3]. The approach in designing SNAP was to find a knowledge representation and a reasoning paradigm useful for natural language processing which exhibits massive parallelism. We have selected marker-passing on semantic networks as a way to represent and process linguistic knowledge.

The work for MUC-5 started at the end of January 1993. Prior to this we had implemented on SNAP a memory-based parsing system which was also used for MUC-4. Since the domain has been changed from MUC-4, we improved the dictionary with semantic tags necessary for EJV domain, developed a new template generation module, and constructed a new knowledge base including concept hierarchy and concept sequence patterns for parsing. Our group consisting of one faculty and five graduate students spent approximately 6 months to engineer a large system.

Approach

The underlying ideas of the SNAP natural language processing system are: (1) memory based parsing, and (2) marker-passing on semantic networks [5]. In SNAP, parsing becomes a guided search over the knowledge base which stores linguistic information. Input words activate and predict concepts in the knowledge base. While the semantic network represents the static part of the knowledge base, marker passing is the mechanism which changes the state of the knowledge base with each new word. Based on the MUC-4 experience, we emphasized efficiency and practicality in developing the system for MUC-5. Full analysis of input texts is avoided. The system picks up only necessary and relevant information from input texts. The knowledge base of domain specific phrasal patterns for TIE-UP relationship has been constructed automatically by using a lexical acquisition system called PALKA, and couple of back-up routes have been established to extract partial information when parsing fails.

SYSTEM ARCHITECTURE

The SNAP system architecture is shown in Figure 1. It consists of a preprocessor, phrasal parser, memory-based parser, knowledge base, and template generator. The preprocessor provides lexical look-up and semantic tagging. The phrasal parser performs grouping of words, and the memory-based parser generates semantic meaning representation by using stored patterns in the knowledge base. The knowledge base consists of a domain concept hierarchy including location hierarchy, a set of phrasal patterns for TIE-UP, and semantically relaxed versions of some phrasal patterns. The phrasal patterns are automatically extracted

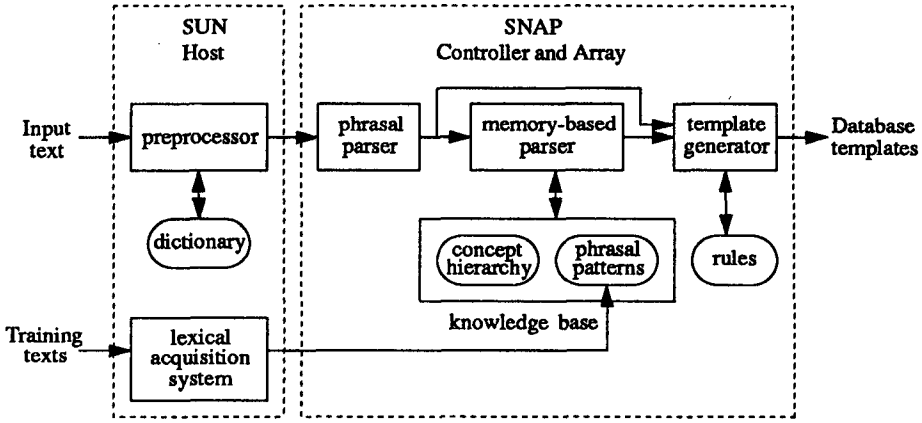


Figure 1: The SNAP information extraction system.

from training texts by the PALKA lexical acquisition system. The template generator has a set of rules to fill slots from the phrasal parser results and the memory-based parser results. To detect a TIE-UP relationship, the template generator first checks the memory-based parser output for the regular patterns. If there is no memory-based parser output, it tries to apply rules to the phrasal parser output, and if no rules are applicable, it checks the memory-based parser output for the relaxed patterns. Figure 2 shows the flow of control including the back-up routes.

PREPROCESSOR

Description of Operation

Our preprocessor is not significantly different from the "generic preprocessor". It locates sentence boundaries and inserts sentence boundary markers for the phrasal parser. In addition, sentences are joined together on one line so that pattern matching can take place for concatenating noun phrases together. Certain noun phrases are grouped together to reduce work load on the phrasal parser. These noun phrases come from 4 different sources: our permuted list of company names, locations from the gazeteer, a list of special words related to facilities, titles, and positions, specifically chosen for filling template slots and/or recognizing the possibility of filling slots, and noun phrases from WordNet. (Company names are a special case that will be discussed below.)

As in our system for MUC-4 last year, our preprocessor continues to group semi-auxiliary verbs like USED_TO and prepositional phrases like SUCH_AS and AS_TO together. Contractions such as CAN'T and DON'T are expanded into CAN NOT and DO NOT. Numbers in the text are normalized by removing commas and/or dollar signs. Abbreviations immediately following numbers without intervening spaces are separated from the number, and expanded; for example 2.3BN is converted into 2.3 BILLION. Three such abbreviations following numbers deserve a small note. When S follows a number, it is always a plural, as in 1980S or IBM MODEL 55S; this S is just removed as no important information required in our system is lost. The abbreviation M is ambiguous, and could mean either million or meters; because it almost always means million in this domain, it is always changed to million. Finally, the obvious abbreviation 4WD is converted to FOUR WHEEL DRIVE. Punctuation markers are separated from the words they are adjacent to, and then each word is looked up in our dictionary. For each word, every possible part of speech is passed on to the phrasal parser segment of our system. In the dictionary look-up function, numbers are a special case. In order that numbers which are not spelled out do not have to be added to the dictionary, the dictionary look-up program tests words to find out if they are cardinal, ordinal, or real numbers, and then reports their part of speech as such. No spelling correction is done in the preprocessor; misspelled words are reported as unknown to the phrasal parser, and they will usually be interpreted as a corporation name if some other word in the noun

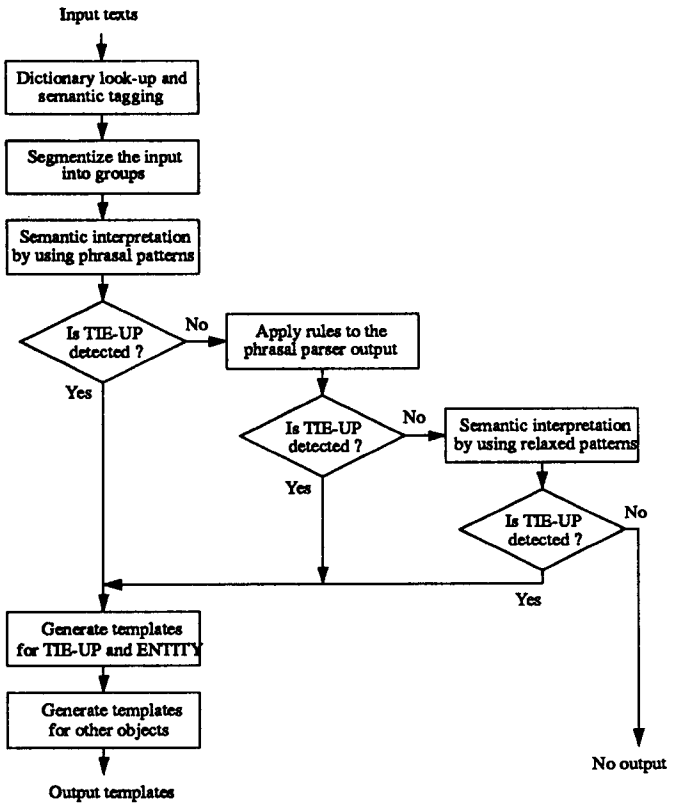


Figure 2: The flow of control in SNAP system.

phrase such as CORP. suggests this, or a human name otherwise. Morphological processing is performed, but not at run time; a set of programs is applied to the input files used in construction of the dictionary to generate all possible inflected forms of words.

As mentioned above, corporation names are recognized and grouped together in the preprocessor. In order to match as many company names as possible, a program was written to generate permutations of the company names in our list of such names. Abbreviations such as INC. are permuted into both INC and INCORPORATED, and abbreviations such as "S.A." are permuted into their various forms such as "S.A" and "S. A.". This increases the size of our list of corporations several-fold. In addition, each of these abbreviations is added to our preprocessor dictionary, and tagged as a corporation name in case only the abbreviation is recognized in some input text.

In addition to the part of speech, root form, and other related information such as tense or number which was in our preprocessor dictionary for passing along to the phrasal parser, many of the words in our preprocessor dictionary were tagged according to the requirements of our rule-based inferencing and template filling module. These tags included MUC5-COMPANY, approximately 12 different types of facilities (MUC5-FACILITY1 for broadcasting stations and studios, MUC5-FACILITY2 for plants, mills, and development facilities, etc.), ten or so different position tags (MUC5-POSITION1 for CEO, chief executive officer, etc., MUC5-POSITION2 for chairman, chairmen, chairman of the board, etc.), and several different human name and title tags (MUC5-NAME-TITLE for Mr., Mrs., etc., MUC5-HUMAN-NAME-FAMILY, MUC5-HUMAN-NAME-GIVEN, MUC5-HUMAN-NAME-MALE, and MUC5-HUMAN-NAME-FEMALE). These tags alerted later processing states that the words so tagged might be important for template filling. Tags were not the only method used for this. The knowledge-base that we built from WordNet, for example, provided a very easy way to recognize words that were part of a date, or words which indicated time. We also built a knowledge-base of location names and possible types (cities, countries, etc.),

from the available gazeteer file, and included the tag MUC5-LOCATION on all words in our preprocessor dictionary which might have been location names in the text. For several pronouns and prepositions such as I and TO, which are also city names in Indonesia, our phrasal parser always ignored this tag. Even though the gazeteer was very large, there are place names missing such as KAOHSIUNG in the walk-through message.

This year, WordNet was used to substantially increase the size of our dictionary. However, through an unfortunate oversight, the morphological programs were not run on the verbs from WordNet in order to generate all of their inflected forms. This caused the word ENTRUSTING to be missing from our dictionary, and not recognized, when it should have been there. Even with WordNet, though, our dictionary doesn't contain all inflected forms of all words. WordNet contains the noun CAPITAL, but not the verb form CAPITALIZE, and so none of the inflected forms of that particular verb were included in our dictionary, and we failed to recognize that CAPITALIZED was either a past tense or past participle verb form. Since our system doesn't do any morphological processing at run time, this is an area where significant improvement could be made with relatively little effort; if unknown words were run through a process that guessed at their part of speech with decent accuracy, then our phrasal parser would do a better job of segmenting each sentence into its constituent phrases. This would have made a difference this year, as there were several times where a verb was unknown, treated incorrectly as a noun by our phrasal parser, and lumped into a noun phrase as part of a corporation name.

Example of Operation

After all stages of preprocessing, except the last—dictionary lookup, the first sentence is all on one line (which can't be seen here), and appears as follows:

```
</SO> begin_text <TXT> 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 .
```

Notice that the period on CO. was converted to an underscore character, the words JOINT_VENTURE TRADING_HOUSE and GOLF_CLUBS were grouped together, and punctuation was split off from the words it was attached to. After the final stage, the data sent to the phrasal parser portion of our system appears as follows:

```
</SO>
0
begin_text
0
<TXT>
0
BRIDGESTONE
1
proper-noun BRIDGESTONE sg MUC5-COMPANY
SPORTS
2
common-noun sport pl MUC5-PROD-SERV
common-noun sports sg MUC5-PROD-SERV
CO_
1
proper-noun CO_ sg MUC5-COMPANY
SAID
2
proper-noun Said sg MUC5-HUMAN-NAME-GIVEN
verb say pp1
...

SET
3
common-noun set sg
proper-noun SET sg MUC5-LOCATION
verb set pp3 n3sg
UP
```

```
2
preposition up
proper-noun UP sg MUC5-LOCATION
A
2
article a sg indefinite
proper-noun A sg MUC5-LOCATION
JOINT_VENTURE
1
common-noun joint_venture sg
...

JAPANESE
2
adjective Japanese
proper-noun Japanese sg
TRADING_HOUSE
1
proper-noun TRADING_HOUSE sg MUC5-COMPANY
...

JAPAN
2
proper-noun Japan sg MUC5-HUMAN-NAME-GIVEN
proper-noun Japan sg MUC5-LOCATION
.
1
punct .
end_sentence
0
```

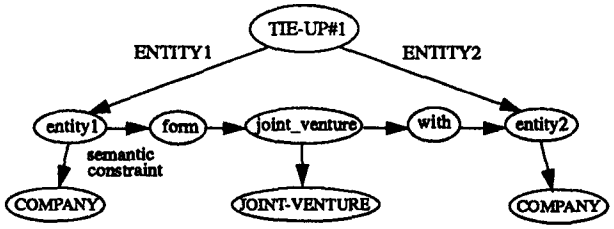
KNOWLEDGE BASE FOR PARSING

The knowledge base consisting of concept nodes and links between them is distributed over the processor array. The parsing is performed within the knowledge base by propagating markers through the network. The knowledge base contains a concept hierarchy, basic concept sequences, and domain specific patterns for TIE-UPs.

Concept Hierarchy and Basic Concept Sequences

Several different sources were used in constructing the knowledge-base used in memory based parsing for this year's system. WordNet was used in the construction of the noun-based ontologically structured knowledge-base, and also in the construction of the verb-based basic concept sequence knowledge-base. The ontological and taxonomical classifications built into WordNet are carried over into our knowledge-base. The English gazeteer was used in building the location knowledge-base. As in WordNet, the inherent structure built into the gazeteer is carried over into the location knowledge-base.

In all of these knowledge-bases, a word may have several different semantic meanings. Our system provided for the representation of these various different meanings, but not the resolution of the semantic lexical ambiguity yet. In the future, new knowledge in the form of appropriate links between the noun-based ontologically structured knowledge-base and the verb-based concept sequence knowledge-base, and semantic priming links between appropriate semantic meaning nodes, will aid in the automatic semantical lexical ambiguity resolution of some of the nouns and verbs. This will occur during parsing, with no extra effort needed by the parser; the knowledge built into the knowledge-bases will guide the markers during marker passing to select the appropriate senses of some ambiguous words. For those words which cannot be disambiguated in this fashion, common-sense and pragmatic inferencing rules can be applied after parsing. Location names, too, can be semantically ambiguous. An extra algorithm of marker passing along a limited number and type of links in the location knowledge-base will be added to the system in the future in order to resolve the ambiguities in location names.



Text form - TIE-UP: [(ENTITY1: COMPANY) form (joint-venture) with (ENTITY2: COMPANY)]

Figure 3: The FP-structure representation

In each knowledge-base, a common structure emerges. Nodes are basically of two types. Some nodes represent the actual word itself, while other nodes represent a possible meaning of a single word or multiple words. Using the terminology of WordNet, we call these "meaning" nodes "synset nodes", and we call the nodes representing the words themselves "word nodes". Except in the case of antonyms, the semantic relationships such as hypernomy, hyponomy, meronomy and holonymy exist as pointers between the synset nodes, because they are relationships between the meanings, and not between the words themselves. For antonyms, however, the antonym relationship exists between individual words and not the meanings of those words. In the future, the hypernym links will be used like ISA links in propagating markers upwards from the nouns, through the restrictions which shall be added to the knowledge-base for semantic ambiguity resolution, to the verb-based concept sequences. Likewise, the meronym and holonym links can be used both for semantic priming, and also for "understanding" sentences where meronymic or holonymic metaphor is used.

Domain Specific Patterns for TIE-UPs

The most important task for the Joint Venture domain is to detect a TIE-UP relationship between several ENTITIES. Since the TIE-UP relationship is represented by using various kinds of expressions, a large amount of rules are necessary to detect the TIE-UPs, after the syntactic and semantic processing. For example, a TIE-UP relationship between company A and company B can be expressed by using the verb *set*, *form*, or *establish* as "A set up joint venture with B", or using the verb *agree* or *sign* as "A agree with B to start joint venture". A verb-oriented semantic representation is not sufficient; one more step is needed to generate the final representation.

This year, we developed a domain specific semantic pattern representation for efficient information extraction. A pattern is represented as a pair of a meaning frame defining the necessary information to be extracted, and a phrasal pattern describing the surface syntactic ordering. This representation is called an *FP-structure*. The FP-structures are used by the parser to detect TIE-UP relationships by pattern matching. Figure 3 shows an example of the FP-structure for TIE-UP relationship. The frame has slots of ENTITY0 (joint venture company), ENTITY1 and ENTITY2, and each slot has semantic constraint MUC5-COMPANY. In the text representation of the FP-structure, the elements with parenthesis are concepts, and other elements are all lexical elements. When the parser performs pattern matching, all the lexical entries, that are mapped under a concept specified in the FP-structure, are matched to their corresponding element. By using the FP-structure, the direct mapping from a surface phrase pattern of input sentence to the target representation of a TIE-UP relation between companies is possible.

Currently, more than 300 patterns for the TIE-UP relationship have been automatically generated by using our lexical acquisition system, and those patterns were successfully used by parser to detect TIE-UPs. The lexical acquisition system, and the performance improvement according to the FP-structures, are presented in the later sections.

PHRASAL PARSER

Modified Input Sentence S1 (Preprocessor Output):

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.

Phrasal Parser Output:

[BRIDGESTONE SPORTS CO_]$_{\text{NOUN-GROUP (CO)}}$ [SAID]$_{\text{VERB-GROUP}}$ [FRIDAY]$_{\text{DATE-TIME-GROUP}}$ [IT]$_{\text{NOUN-GROUP}}$
[HAS SET]$_{\text{VERB-GROUP}}$ [UP]$_{\text{PREPOSITION}}$ [A JOINT_VENTURE]$_{\text{NOUN-GROUP (JV)}}$ [IN]$_{\text{PREPOSITION}}$
[TAIWAN]$_{\text{NOUN-GROUP (LO: COUNTRY)}}$ [WITH]$_{\text{PREPOSITION}}$ [A LOCAL CONCERN]$_{\text{NOUN-GROUP (CX)}}$ [AND]$_{\text{CONJUNCTION}}$
[A JAPANESE TRADING_HOUSE]$_{\text{NOUN-GROUP (CO)}}$ [TO]$_{\text{PREPOSITION}}$ [PRODUCE]$_{\text{VERB-GROUP}}$ [GOLF_CLUBS]$_{\text{NOUN-GROUP}}$
[TO]$_{\text{PREPOSITION}}$ [BE SHIPPED]$_{\text{VERB-GROUP}}$ [TO]$_{\text{PREPOSITION}}$ [JAPAN]$_{\text{NOUN-GROUP (LO: COUNTRY)}}$ [.]$_{\text{PUNCTUATION}}$

Relevant Concept Sequences in the Knowledge Base:

TIE-UP186: [(ENTITY1: MUC5-COMPANY) say it set up (joint-venture) with (ENTITY2: MUC5-COMPANY) and (ENTITY3: MUC5-COMPANY)]
TIE-UP164: [(ENTITY1: MUC5-COMPANY) say it set up (joint-venture) with (ENTITY2: MUC5-COMPANY)]
TIE-UP185: [(ENTITY1: MUC5-COMPANY) set up (joint-venture) with (ENTITY2: MUC5-COMPANY) and (ENTITY3: MUC5-COMPANY)]
TIE-UP165: [(ENTITY1: MUC5-COMPANY) set up (joint-venture) with (ENTITY2: MUC5-COMPANY)]
TIE-UP66: [(joint-venture) with (ENTITY1: MUC5-COMPANY) and (ENTITY2: MUC5-COMPANY)]

Memory-Based Parser Output:

(TIE-UP186#1 (ENTITY1: BRIDGESTONE SPORTS CO_)
(ENTITY2: A LOCAL CONCERN)
(ENTITY3: A JAPANESE TRADING_HOUSE))

Figure 4: Outputs of preprocessor, phrasal parser and memory-based parser

From a sequence of dictionary definitions of entries in the modified input sentence, the phrasal parser produces a sequence of phrasal segments consisting of relevant entries and assigns features to noun groups which will be used by the template generator.

Phrasal Segmentation

The list of phrasal segments is as follows: noun group, verb group, adverb group, date time group, and several important word classes such as conjunction, preposition, relative pronoun, and punctuation, and the word "that" and possessive marker ('s). In case of syntactic ambiguity which still exists after ambiguity resolution based on local syntactic and semantic information, the phrasal parser returns several phrasal segments where the input text can be parsed into different phrases. Figure 4 shows a sequence of phrasal segments generated from the first sentence S1 of the walkthrough message.

We describe below how these phrasal segments are produced from the preprocessor outputs using a noun group example. Noun group refers to the noun phrase up to the head noun. Qualifiers of a head noun such as prepositional phrases and relative clauses are processed separately. Noun groups have an intricate syntactic structure. However, the ordering of constituents in the noun group is relatively fixed as shown below:

pre-determiner + article + quantifier + demonstrative pronoun +
possessive pronoun + numerical modifier + ordinal + cardinal +
adverb* + (adjective* | past-participle | present-participle) + noun*

Most of the choices are realized by including or omitting possible constituents. An asterisk indicates that a single slot can be filled by a sequence of constituents of the same type. The ordering constraint is used to capture the possible syntactic patterns of noun groups by a straightforward mapping of slot and filler analysis as shown below:

the result → article + common-noun
a densely populated area → article + past-participle + common-noun
its 1985 takeover → possessive-pronoun + ordinal + common-noun

1. Place a P marker on the CSR and its first CSE of each concept sequence.

2. Get an input phrase from the phrasal parser.

3. Generate a concept instance from the input phrase.

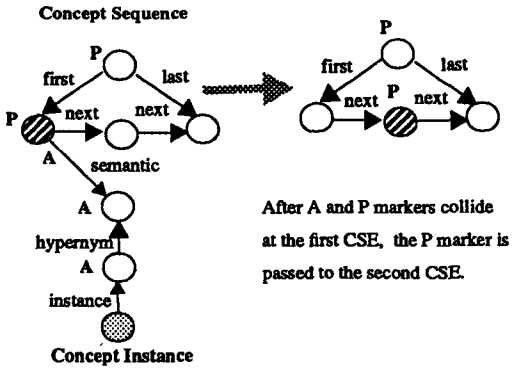
4. Propagate A markers from the concept instance to the corresponding CSEs through a chain of instance, hypernym, and reverse-semantic links.

5. If a P-marked CSE receives an A marker, accept the CSE. Then pass the P marker to the next CSE through next link, or propagate an A marker to the CSR through reverse-last limk.

6. If a P-marked CSR receives an A marker,accept the concept sequence and then propagate Cancel markers from the CSR through inhibition link to cancel the subsumed concept sequences which are also accepted.

7. If all input phrases are processed, generate CSIs of the accepted concept sequences as interpretation of the input sentence. Else, go to 2.

(a) Basic marker-passing steps



(b) Example of A and P marker collision

Figure 5: Baseline memory-based parsing algorithm

Although this noun group syntactic structure is still a simplification compared to the full range of English noun group constructs, it covers a wide variety of noun groups. In addition, whenever an exception occurs, a new noun group pattern covering the exception is added to the set of existing noun group patterns. In this way, we have successfully developed an extensive set of noun group patterns by using the MUC-5 test corpus.

Noun Group Feature Assignment

The noun group features assigned by the phrasal parser are as follows:

CO	MUC5 company
CX	non-specific reference to a company
CS	non-specific reference to a group of companies
GO	government agency
PE	person name (FAMILY, GIVEN, MALE and FEMALE names)
JV	reference to joint venture
LO	location (CITY, PROVINCE, and COUNTRY)
PO	company position
IN	industry type
PS	product or service
FA	facility
TI	time or date
MO	monetary reference
NU	numerical reference
PC	percentage symbol
CP	capitalized word

MEMORY-BASED PARSER

The memory-based parser matches the sequence of phrasal segments with concept sequences stored in the knowledge base via a series of parallel marker-passing commands. When a concept sequence for a tie-up relation is indexed by a set of input phrasal patterns, it generates a pseudo-template. From this output, the template generator produces a template on tie-up relations.

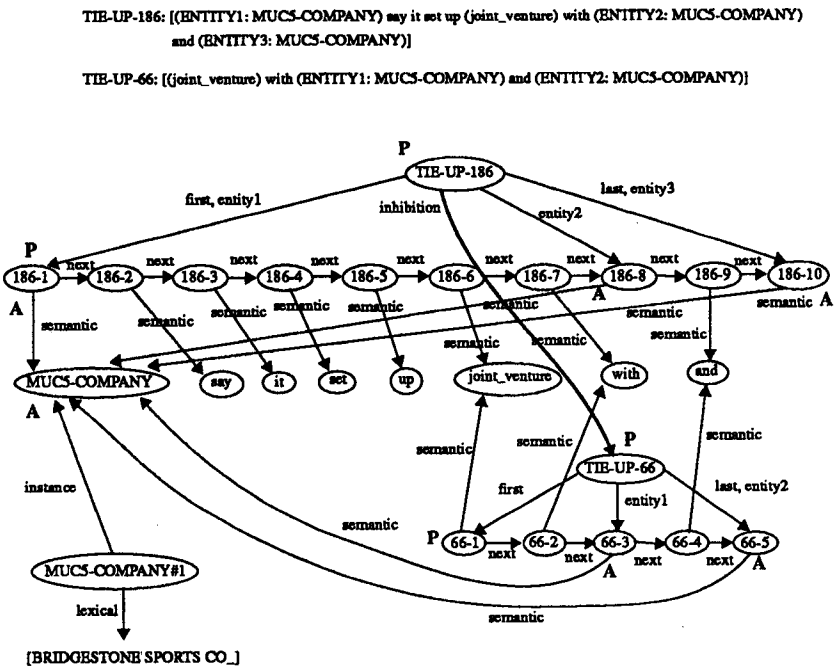


Figure 6: Marker state of the sample knowledge base after the first input phrase of S1 is processed

Memory-Based Parsing Algorithm

The memory-based parsing algorithm is based on repeated applications of top-down predictions and bottom-up activations by propagating markers in parallel throughout the semantic network knowledge base. Figure 5-(a) shows the basic marker-passing steps used to match an input sentence with relevant concept sequences. Prediction-markers (P markers) are used to predict some concept sequence elements as potential hypotheses of the input phrase. Activation-markers (A markers) are used to propagate activations from the input phrase to the corresponding concept sequence elements. As shown in Figure 5-(b), among the P-marked concept sequence elements, those which receive A markers are accepted as valid hypotheses. After a P-marked concept sequence element is accepted, the P marker is passed to the next concept sequence element and waiting to be activated by the next input phrase. When all concept sequence elements of a concept sequence are accepted, a concept sequence instance is generated for meaning representation.

Parsing Example

We describe here how the memory-based parser produces a pseudo-template from an example sentence S1 of the walkthrough message. At the start of parsing, all concept sequences are potential hypotheses for an incoming sentence. Therefore, in step 1, all concept sequences are initially predicted by placing P markers on their CSRs and the first CSEs. For example, in the example knowledge base shown in Figure 6, P markers are placed on `TIE-UP-186`, `186-1`, `TIE-UP-66` and `66-1`. In step 4, from the first input phrase, `[BRIDGESTONE SPORTS CO_]`, a concept instance `MUC5-COMPANY#1` is generated and connected to the corresponding semantic concept `MUC5-COMPANY` via `instance` link, as indicated in Figure 6. These links are used as paths for propagating bottom-up activations from the concept instance `MUC5-COMPANY#1` to the corresponding concept sequence elements in step 5. Even though the CSEs `186-1`, `186-8`, `186-10`, `66-3` and `66-5` in Figure 6 received A markers, only the P-marked CSE `186-1` is accepted, and therefore, `MUC5-COMPANY#1` is bound to `186-1` via `cse-instance` link, as shown in Figure 7. A markers at other CSE nodes are disregarded, because they are not predicted at this stage. As a result, the P marker at `186-1` is passed to the next CSE `186-2`. However, other P markers are not affected by this change. The P-marked nodes (in this case, `186-2`, `66-1`, `TIE-UP-186` and `TIE-UP-66`) again wait for A markers generated from the next input phrase.

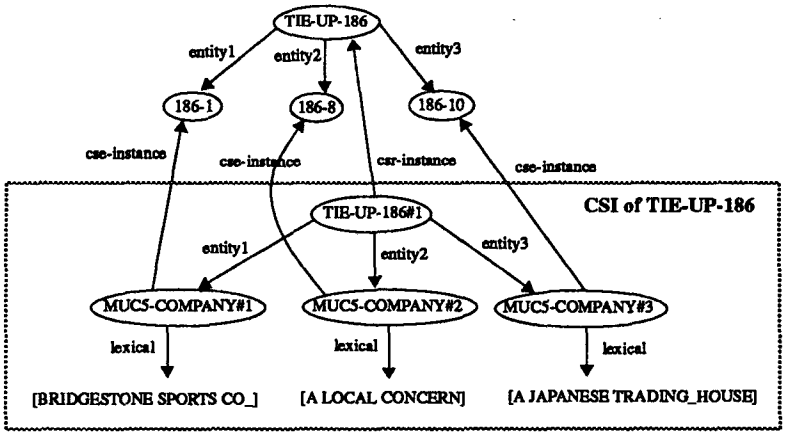


Figure 7: Interpretation of S1

This procedure is repeated until all input phrasal segments are processed. Even though many nodes are predicted in the early processing stage, as more input words are processed, only a few nodes receive activations and are accepted. Eventually only relevant hypotheses survive as the interpretation of the input sentence. In this way, a parsing problem is converted into a phrasal pattern matching problem. The memory-based parser acts as a filter, in which each new input word reduces the number of possible meanings.

TEMPLATE GENERATION MODULE

Overview

The template generation module is a rule-based system, which consists of 4 levels as shown in Figure 8. The first part generates templates based on the output of the memory-based parser with strict constraints for the verb cases. The second part generates templates based on the output of the phrasal parser, when the memory-based parsing failed to generate correct interpretation. The third part is almost same as the second part, but it requires less strict string patterns to apply the rules. The forth part is also based on the memory-based parsing output but with less strict verb case constraints. When applying the rules, latter parts are not applied when a former part succeeds in generating templates.

Template Generation Based on Memory-Based Parser

When memory-based parsing is successful, generated concept sequence instances are marked. In the template generation process, some markers are propagated from this marker, and the information is extracted by collecting markers. The memory-based parser not only gives the syntactic information but also gives the semantic interpretation. Thus, the output contains the template generation rules by itself, and the templates generated have high precision rate. All concept sequences are attained by an automatic knowledge acquisition tool from the TIPSTER texts and keys. There are about 300 concept sequences for template generation in the knowledge base.

One major problem is caused by the advantage above, because concept sequences require the satisfaction of strict semantic constraints. Because many company and human names are not known to the system and there is no inference routine to guess company or human names, the memory-based parser fails on many sentences that are matched to the existing concept sequences but doesn't satisfy the semantic constraints. To overcome this problem, the semantic constraints of some part of the concept sequences are relaxed to nothing. These concept sequences with relaxed semantic constraints are chosen very carefully from the concept sequences with semantic constraints, in order to guarantee correct template generation. About 100

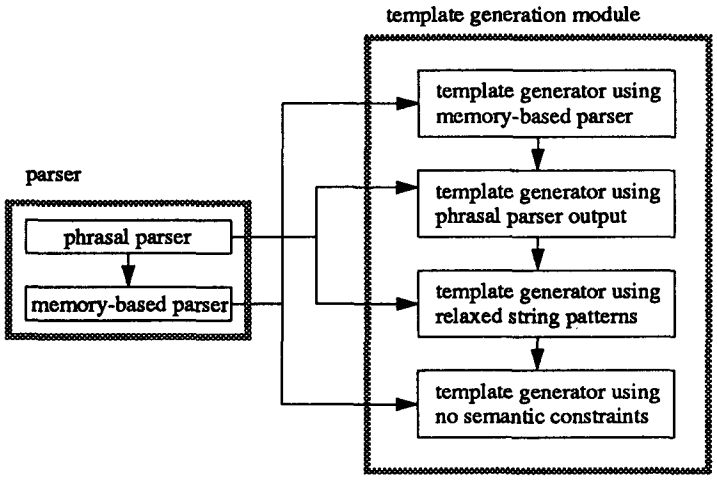


Figure 8: Template generation module

concept sequences were used for the final run. Both of the modules with or without semantic constraints only give the template information about *tie-up relationship*, *entities* and *entity relationship*. Other template information such as *activity*, *industry* and so on is filled out based on phrasal parser output.

Template Generation Based on Phrasal Parser

In this part, information is extracted by matching string patterns. Important noun groups are classified using a special data structure, which specifies if a noun phrase has any (or multiple) of *company name*, *special common nouns* (e.g. venture, company or firm), *human name*, *nationality*, *city name*, *product name*, *facility* and so on. The template generation module uses actual string input and data structure, and extracts the required information when there is a given pattern which matches the text input.

Though the phrasal parser does not give the semantic interpretation, the success rate of phrasal parser is much higher than the memory-based parser, and it is used as a back-up input for the template generation module. Because it doesn't have any semantic information, the given strict string patterns impose a kind of semantic constraint in template generation. For the purposes of template generation based on the phrasal parser, domain-specific rules were generated. These rules form the skeleton of the template generation module and serve as the driving force behind the template generation process. All template generation based on the phrasal parser is due to both the successful application of a specific rule, and the implementation of the actions associated with the specific rule. Another use of the rules was as an aid in building concept sequences for the knowledge base to be utilized by the memory- based parser.

Rule generation was performed through the use of training sentences obtained from the TIPSTER texts and also from human intuition. Training sentences were analyzed for important concepts, words, punctuation, symbols, etc. (e.g. company name, 'with', '$', ','). Sequential patterns of these targeted attributes were then formed. This process was repeated for all template fill categories. Around 750 rules (patterns) are implemented to be used to extract template information about tie-up relationship, entity, entity relationship, activity, industry, product/service, etc.

Application of the rules is dependent upon the identification of concepts found to be important for the joint-venture domain. The identification of important concepts for use in rule application by the template generation module is performed by the preprocessor and phrasal parser. The preprocessor identifies important domain-specific concepts through the use of a dictionary containing lists of words – some of which are very long (e.g. company names, locations). Several files made available to MUC5 participants were utilized in forming these lists. The phrasal parser performs syntactic analysis in order to disambiguate results pro-

modules	used parsing output	constraints	number of rules
memory-based template generation	memory-based parser	semantic constraints for verb cases	300
phrasal template generation	phrasal parser	strict string patterns	750
relaxed phrasal template generation	phrasal parser	less strict string patterns	3
relaxed memory-based template generation	memory-based parser	no semantic constraints for verb cases	100

Table 1: Description of template generation modules

duced by the preprocessor.

Results obtained by the application of rules by the template generation module based on the phrasal parser augment the results obtained from the memory-based parser. Although the memory-based parser produces precise results, it requires a highly formalized structure for its knowledge base. Due to size limitations and time limitations regarding construction of the knowledge base, a knowledge base of sufficient size and complexity was not available to the memory-based parser – therefore making it more prone to failure. The precision of results obtained through rule application was not as high as the precision of results obtained by utilizing the memory-based parser. However, since the structural requirements for rules in this system are not as highly formalized, and since size limitations are not as severe, the likelihood of failure is substantially reduced.

Definite Reference Resolution

The activation-based reference resolution scheme which has been developed as a part of SNAP project [1] could not be used, because it relies on the memory-based parsing output. Therefore, a very simple reference resolution scheme is used with phrasal parser output, only for the joint venture and the entities. When a definite reference to a joint venture (e.g. "the joint venture") is found, the most recent joint venture is chosen as a referent. The same scheme is applied to definite references to entities, except the cases using company names (e.g. aliases). A small module has been written to handle acronyms and company aliases.

LEXICAL ACQUISITION SYSTEM

To provide the domain specific semantic patterns (FP-structures) presented in the previous section, an automatic lexical acquisition system called PALKA has been developed [4]. The major goal of this system is to facilitate the construction of a large knowledge base of semantic patterns. PALKA acquires semantic patterns from a set of domain specific sample texts.

The information extraction task on a narrow domain is quite different from a general semantic interpretation task. First, there are only a small number of event categories to which each text should be mapped, and only a small amount of pre-defined types of information to be extracted. Various expressions need to be mapped into one event category. Second, the information to be extracted can be found anywhere in the sentence, not only in the subject or the object of the sentence, but also in the prepositional phrases or in the modifier. An efficient representation should map various expressions to one of the desired categories, and detect the information carrying words or phrases from anywhere in the sentence. Based on these observations, the FP-structure has been developed as a suitable representation.

Acquisition Procedure

First, a meaning frame including several keywords is defined as a target of the acquisition. The TIE-UP meaning frame contains ENTITY0, ENTITY1, and ENTITY2 slots, each of which has its semantic constraint as MUC5-COMPANY. ENTITY0 corresponds to the joint venture company (child company). The

KYOCERA AGREES TO FORM JOINT_VENTURE FOR FINANCING AND LEASING WITH SANWA BANK. TIE-UP: [(ENTITY1: COMPANY) agree to form (JOINT_VENTURE) with (ENTITY2: COMPANY)]
CIBA-GEIGY LIMITED OF BASEL, SWITZERLAND, AND CIBA-GEIGY CORPORATION HAVE SET UP A JOINT_VENTURE IN MICROELECTRONIC MATERIALS. TIE-UP: [(ENTITY1: COMPANY) and (ENTITY2: COMPANY) set up (JOINT_VENTURE)]
MITSUBISHI CORP_ IS ENTERING INTO A JOINT_VENTURE WITH THE AYALA CORP_ TO CONVERT 300 HECTARES OF REAL_ESTATE IN LAGUNA PROVINCE INTO ANOTHER INDUSTRIAL PARK. TIE-UP: [(ENTITY1: COMPANY) enter into (JOINT_VENTURE) with (ENTITY2: COMPANY)]
CINCINNATI BELL INFORMATION SYSTEMS FORMED A JOINT_VENTURE WITH KINGSTON COMMUNICATIONS PLC TO MARKET SOFTWARE PRODUCTS AND SERVICES IN EUROPE. TIE-UP: [(ENTITY1: COMPANY) form (JOINT_VENTURE) with (ENTITY2: COMPANY)]

Figure 9: The example FP-structures

word "joint venture" has been selected as a keyword. Relevant sentences are extracted from training texts by using the keyword. Second, an extracted sentence is segmentized and decomposed into several simple phrasal patterns. The original text consists of complex sentences which contain relative clauses, nominal clauses, conjunctive clauses, etc. Since semantic patterns are acquired from simple clauses, it is necessary to convert a complex sentence to a set of simple clauses. The phrasal parser groups words based on each word's syntactic category and ordering rules for noun-groups and verb-groups. After grouping is performed, the phrasal parser first simplifies the sentence by eliminating several unnecessary elements such as determiners, adverbs, quotations, brackets and so on. Then it converts the simplified sentence into several simple clauses by using conversion rules. The conversion rules include separation of relative clauses, nominal clauses and conjunctive clauses. Third, by using the semantic tags provided by the preprocessor, links between the frame slots and the phrasal pattern elements are established. All the noun groups representing a company name are mapped as an ENTITY. Separating ENTITY0 from others is done by a human after generating all the FP-structures. Finally, PALKA constructs an FP-structure based on the mapping information. The basic strategy for constructing an FP-structure is to include the mapped elements and the main verb, and discard the unmapped elements.

Figure 9 shows several examples of the training sentences and the FP-structure generated from them for TIE-UP. As discussed in the performance analysis section, the FP-structures generated by PALKA greatly affects the overall scores. Also by automating the procedure of building patterns, it reduces the time for constructing the knowledge base of semantic patterns, and provides scalability and portability to the knowledge based information extraction.

PERFORMANCE AND DISCUSSION

Summary of Score Results

During the development of the system, the 186 dryrun messages were used as a test set. For the final run, 251 new messages were processed. A part of the summary of score results for the final run is shown in Tables 2 and 3. In Error-based metrics, ERR = 84%, and in recall-precision-based metrics, the recall for all objects was 19%, the precision was 36%, and the F-measure was 24.67.

While developing the system, we have tried various options to improve the performance. The most important option was changing the semantic pattern knowledge base. To detect the TIE-UP relationship, we used simple string matching rules, domain specific FP-structures, and a relaxed version of the FP-structures. Table 4 shows the different results according to the selection of different knowledge base options.

The basic system only include the simple string matching rules. The string matching rules are too primitive, so the recall was only 10%. With the FP-structures, the recall improved 9%, the precision decreased

SLOT	POS	ACT	COR	PAR	INC	ERR	UND	OVG	SUB
ALL OBJECTS	12076	6208	2220	71	1676	84	67	35	43
MATCHED ONLY	4406	3286	2220	71	660	52	33	9	24

Table 2: Summary of score results - Error-based metrics.

SLOT	POS	ACT	COR	PAR	INC	REC	PRE	UND	OVG
ALL OBJECTS	12076	6208	2220	71	1676	19	36	67	35
MATCHED ONLY	4406	3286	2220	71	660	51	69	33	9
TEXT FILTERING	251	180	173	-	-	69	96	31	4
				P&R		2P&R			P&2R
F-MEASURES				24.67		30.56			20.69

Table 3: Summary of score results - Recall-precision-based metrics.

slightly, and the overall F-measure increased more than 8% The third row is the result when relaxed patterns are included. In the relaxed pattern, the semantic constraints are removed, so any noun group can be mapped to an ENTITY slot. The relaxed patterns were tested because there were many company names that were not detected (they are not in the company name list). Due to the relaxed constraint, more TIE-UP relations were detected, so the recall increased 3%. However, the relaxed patterns also produced many incorrect result, and the precision decreased 10%, resulting in a slightly decreased F-measure. By carefully selecting only part of the relaxed pattern, the overall performance would increase by detecting unknown company names without producing a large amount of incorrect matching.

Amount of Effort

The development of whole system including the knowledge base took 6 month in our team consisting of one faculty and five graduate students. Since the skeleton of the memory-based parser used for MUC-4 was reusable, most of the time was spent on construction of the knowledge base, updating the dictionary entries, and coding the template generation rules. The total effort for MUC-5 can be estimated as follows:

Knowledge base construction	25%
Preprocessor and dictionary	20%
Memory based parser	10%
Template generation	25%
System integration	10%
Testing and scoring	10%

Limiting Factors

Regarding the limiting factors in the performance of the system we have noticed that: (1) Inferencing rules to detect unknown company names are needed, (2) More semantic patterns are needed, (3) Our discourse processing capability was insufficient, (4) The parser does not address enough linguistic problems, and (5) The semantic tagging in dictionary entries was not complete.

Strengths and Weaknesses

CONFIGURATION	REC	PRE	F-MEASURE
The basic system	10	38	16.19
With FP-structures	19	36	24.67
With FP-structures and relaxed patterns	22	26	23.71

Table 4: Score results with different configurations

The burden to the parser and template generator was greatly reduced by moving a large part of processing into the preprocessor. The preprocessor not only performs the dictionary look-up, but also detects various multi-word nouns, and provides semantic tags for important words, which are necessary for later processing stages. The use of domain specific semantic patterns to detect TIE-UP relationships is efficient and effective. The memory-based parser performs fast and efficient semantic parsing by using those patterns. The lexical acquisition system was effective in constructing the knowledge base of patterns. It also provides portability to other domains. The main weaknesses lie in the lack of discourse processing and lack of a module that performs inference to detect company names. A large portion of the dryrun and finalrun messages did not produce a correct TIE-UP relationship only because we failed to detect the company names.

Reusability

Assuming that the domain and the required output is changed, approximately 75% of the lexicon and concept hierarchy is reusable. None of the inferencing rules for filling templates and the domain specific patterns are reusable. However, the patterns can be easily acquired for a new domain by using the lexical acquisition system.

What was Learned

Through the development of the SNAP system for MUC-5, we have learned that the information extraction task needs different approaches from the Natural Language Understanding task, although a full natural language understanding capability is necessary for perfect performance. Full syntactic analysis and semantic interpretation are not necessary for information extraction. If sufficient semantic information is provided for each lexical entry, a phrasal pattern matching approach can achieve reasonable performance in a very efficient way. Over all, our experience with MUC-5 has been useful and rewarding.

REFERENCES

[1] Cha S. and Moldovan D., "A Marker-Passing Algorithm for Reference Resolution," *Proceedings of IEEE International Conference on Tools with Artificial Intelligence*, 1993.

[2] Chung, M. and Moldovan, D., "Memory-Based Parsing with Integrated Syntactic and Semantic Analysis on SNAP", *Technical Report PKPL 91-10*, Dept. of Electrical Engineering-Systems, University of Southern California, 1991.

[3] DeMara, R. and Moldovan, D., "The SNAP-1 Parallel AI Prototype", *Proceedings of Annual International Symposium on Computer Architecture*, 1991.

[4] Kim, J.-T. and Moldovan, D., "Acquisition of semantic patterns for information extraction from corpora," *Proceedings of CAIA-93, the Ninth IEEE Conference on AI Applications*, 1993.

[5] Moldovan, D., Lee, W., Lin, C., and Chung, M., "SNAP: Parallel Processing Applied to AI", *IEEE Computer*, May 1992.

[6] Riesbeck, C. and Martin, C., "Direct Memory Access Parsing," *Report 354*, Dept. of Computer Science, Yale University, 1985.

[7] Sundheim B., editor, *Proceedings of the Fourth Message Understanding Conference (MUC-4)*, Morgan Kaufmann Publishers, Inc., San Mateo, Ca, June 1992.