Parsing Korean based on Dependency Grammar and GULP

So Young Kwon
PhD Program in Computational Linguistics
The University of Georgia

2003 May 5

Abstract

This paper presents a parsing algorithm in Prolog using GULP, based on dependency grammar and unification-based grammar.[1] It parses declarative sentences of a free-word-order language, Korean. The dependency grammar accepts free order of the words in a sentence. Unification-based features separate the grammar from the parsing algorithm and also simplify the notation of the grammar. GULP (Graph Unification Logic Programming) is an extension to Prolog that facilitates the implementation of unification-based grammar.

1 Introduction

Korean is a partial free-word-order language. Most of the words are free in order, but there are restrictions in some words. For example, verb and verbal endings must always come in the final position of a sentence and adjectives must always precede the modifying noun . Dependency grammar provides a clear description of word order in Korean. Also, if we combine dependency grammar with unification-based grammar, implementation of a Korean parser becomes simple and efficient. By representing a word with features and a grammar with unification of features, the parsing algorithm is separated by the grammar. The Korean parser is implemented in Prolog,

[1]I would like to thank Jonathan McClain for proofreading this paper.

using GULP (Covington 1994), a tool for implementing unification-based grammar.

2 Dependency grammar

Dependency grammar describes sentence structures by linking individual words and specifying their relations. Each link will have a `head` and a `dependent`. In general, the `dependent` is the modifier or complement and the `head` determines the attribute of the `dependent`. Although there are variations in the decision of `head` and `dependent` relations among languages, the ultimate `head` , a `head` that is not a `dependent` of any other `head`, would normally be a single `head` for a single sentence.[2] For example, the dependency relation of the sentence, "The dog chased the cat." is as follows:

(1)
```
chased
   dog
     the
   cat
     the
```

The `dependent` is represented by using indentation, and the ultimate `head` is 'chased'.[3]

3 Unification-based grammar and GULP

Unification-based grammar uses features and their values to represent a grammar. The feature structure plays a prominent role in this grammar. Each feature structure has the information of part of speech, number, gender, tense, subcategorizations, etc. The values of the items in the feature structure are unified (matched or merged) order independently. Unification-based grammar is a form that can be represented in various grammar theories. A

[2] Hudson (1984) argues that sentences with raising verbs have multiple heads. For example, the noun 'John' in 'John seems to like Mary.' has two heads: 'seems' and 'like'.

[3] Covington (1990) presents several different ways to graphically represent the dependency relation. The most common way is to draw an arrow pointing from the head to the dependent. The indentation will be the most convenient representation for the output of the Korean parser.

phrase structure grammar rule can be represented with feature structures, as in (2).

(2)
$$\underset{\begin{bmatrix} tense : T \end{bmatrix}}{S} \rightarrow \underset{\begin{bmatrix} case : nom \\ number : N \end{bmatrix}}{NP} \quad \underset{\begin{bmatrix} tense : T \\ number : N \end{bmatrix}}{VP}$$

The value, `nom` is `nominative`. The features of `NP`, `VP` and `S` are all merged and the order of which feature structure merges does not change the result. A dependency grammar can be represented with feature structures, also.

(3)
```
chased(subcategorization:2)
   dog(case:nom)
     the
   cat(case:acc)
     the
```

The value `acc` is `accusative` in (3).

Unification-based feature structures separate the grammar from the parsing algorithm. The feature structures are notated and unifications are done in the grammar rule. Also, the order in which the parser calls for rules does not matter because unification is order-independent.

Covington (1994) presents an extension to Prolog, GULP (Graph Unification Logic Programming). This software tool facilitates the implementation of unification-based feature structures in Prolog. The syntax of GULP is, for example, `a:b..c:d`, which represents a feature structure of `a` having the value `b` and `c` having the value `d`, and all other features uninstantiated. The interpreter in GULP transfers `a:b..c:d` into Prolog terms and unifies the features. Then by the built-in predicate, `print`, the Prolog terms are translated back to GULP notations.

4 Characteristics of Korean

Kwon and Yoon (1989) pointed out several characteristics of Korean.

First, postpositional function words. They are grouped into two major categories: postpositions for nominative and accusative case markers and postpositions for verbal endings. These postpositional words and also verb stems cannot stand alone in a sentence.

Second, word order. Korean is a partial free-word-order language:

- verb and verbal ending(s) must come in the final position of a sentence, while the words before the verb are free or even omitted (4a,4b).
- postpositional function word must follow right after the word it modifies and an adjective must precede right before the noun it modifies (4c).

(4)

(a)	*gang-a-gi*	*ga*	*go-yang-i*	*lul*	*chot*	*nun-da*	
	dog	nom	cat	acc	chase	verbal	ending
	'The dog chases the cat.'						
(b)	*go-yang-i*	*lul*	*gang-a-gi*	*ga*	*chot*	*nun-da*	
	cat	acc	dog	nom	chase	verbal	ending
	'The dog chases the cat.'						
(c)	*gang-a-gi*	*go-yang-i*	*ga*	*lul*	*chot*	*nun-da*	
	dog	cat	nom	acc	chase	verbal	ending
	ungrammatical sentence						

5 Unification-based dependency parser

The Korean parser uses the dependency parser in Prolog presented by Covington (2003). Several modifications are done to the dependency parser by using GULP and changing the dependency rules and parsing algorithm.

The parser has three sections. First, the lexicon. Second, the dependency rule. Third, the parsing algorithm.

5.1 Lexicon

Each word is represented in the form of

```
word(PhoneticSound, PartofSpeech(GULPFeatures)).
```

where, `PhoneticSound` is the actual input word of the sentence to be parsed. `PartofSpeech` consists of `n` for noun,`v` for verb stem, `pp` for postpositional case marker, `p` for postpositional verbal ending,and `adj` for adjective. `GULPFeatures` are features notated in GULP syntax.

(5) `word(gang-a-gi, n(ending:vow..sem:dog)).`

The lexical entry for the noun, 'dog' is represented in (5). The `ending` feature is a sound agreement between the `head` and the `dependent`: if the end of the letter of the preceding `dependent` is a vowel, then the following `head` must start with a consonant, and vice versa. The value `vow` and `con` is decided by the value of the `dependent`. For example, the noun *gang-a-gi*, the `dependent`, has the value `vow` and the postposition *-ga*, the `head`, has the same value.

5.2 Dependency rules

There are three facts to be stated in each dependency rule. First, the relation of `head` and `dependent`.

- The postpositional function word is the `head` and the modified noun or verb is the `dependent`.
- The verb stem is the `head` and the postpositional case marker is the `dependent`.
- The postpositional verbal ending is the `head` and the verb stem is the `dependent`.

The ultimate `head` is the verbal ending, which always appears in a sentence.

Second, the order restrictions. As mentioned in section 4, a postpositional word and an adjective must be adjacent to their `dependent`, respectively. On the other hand, the postpositional case markers should precede, but not have to be adjacent, to the verb. In order to implement these differences in word order restrictions, once a word goes into the parsing algorithm, the notation of a word changes to a list as follows:

```
[Number, ListofDependents, PhoneticSound,PartofSpeech,GULPFeatures].
```

`Number` is the position of the word in a sentence. `ListofDependents` is the list of all the `dependent(s)` and it's `dependent(s)` and so on. The `Number` is used in the conditions of the dependency rules, to state the order restriction. For example, if the `Number` of the `dependent` is N1 and that of the `head` is N2, the 'adjacency restriction' is stated as '$N2$ is $N1 + 1$' and the 'preceding restriction' as '$N1 > N2$'.

Third, the agreements of features. Agreement of `ending` features are stated in the conditions of the dependency rules.

The dependency rules are represented in the form of

```
check_dh(Dependent,Head) :- Condition.
```

where, `Condition` states the order restrictions and the feature agreement.

(6)
```
check_dh([NumberA,_,_,n,FeatureA],[NumberB,_,_,pp,FeatureB]) :-
          FeatureA = ending:End, FeatureB = ending:End,
          NumberB is NumberA+1.
```

The dependency rule for noun and postpositional case marker is shown in (6). Both the feature agreement and the order restriction are stated in the condition of the rule.

5.3 Parsing algorithm focused on Korean

The input of the parser is a list of words composing the sentence that needs to be parsed. The output would be a notation of each words showing the dependency relation by indentation and also displaying the features in GULP syntax.

(7)
```
?- try([gang-a-gi,ga,go-yang-i,lul,chot,nun-da]).

    6 [nun-da, p, ending:con..tense:pres]
      5    [chot, v, ending:con..sem:chase..subcat:2]
         4       [lul, pp, case:acc..ending:vow]
             3           [go-yang-i, n, ending:vow..sem:cat]
         2       [ga, pp, case:nom..ending:vow]
             1           [gang-a-gi, n, ending:vow..sem:dog]
```

The input and output for sentence 4a is shown in (7).

The parser has roughly 4 steps.[4] First, the parser takes the first word in the input list and looks it up in the lexicon. The lexical entry changes to a list mentioned in 5.2. Second, it looks through the `HeadList`, which is a list of words that seems to be the `head`, to see if there is a `dependent` to the current word. If so, it removes the `dependent` from the `HeadList`. Third, it looks through the `WordList`, which is a list of all the words seen, to see if there is a `head` for the current word. If there is, the parser links them together. Otherwise, it puts the current word in `HeadList`. Fourth, it puts the current word in `WordList`.

[4]See Covington (2003) for detail documentation of the 4 steps.

When the recursive step from the first to the fourth finishes, the ultimate head in the `HeadList` is the postpositional verbal ending. But, there was a restriction in Korean mentioned in section 4, that postpositional function word and verb stem cannot stand alone in a sentence. This can be implemented in the condition of the predicate that terminates the recursion of the parsing algorithm.

(8)
```
parse_loop(_,[],_,[H],[H]):- \+ (H = [_,[],_,pp,_];
                   H = [_,[],_,p,_]; H = [_,[],_,v,_]).
```

Since each word is represented as a list of the form, [Number, ListofDependents, PhoneticSound,PartofSpeech,GULP features], (8) means that if you are in the last step of the recursive parsing algorithm, and the last word does not have any `dependent`, then that word cannot be `pp,p` or `v` : if there is no dependent to a word, in the last step of the algorithm, the word stands alone in a sentence.

The output of the parsing algorithm itself is a Prolog term. The features must be retranslated from a Prolog term to a readable structure. The built-in predicate of the GULP system, `display_feature_structure/1`, displays the feature structure in a readable tabular format.

6 Possible improvements

There are several limitations in the Korean parser.

- The Korean parser can only parse simple declarative sentences. It needs to be improved to parse sentences with subordinate clauses and coordinate clauses.
- The subcategorization feature plays a different role in Korean than in English. For example, if the subcategorization of a verb is 2, this would mean that there can be *at most* two arguments for the verb. This property is not stated in the dependency rule of the Korean parser.

These limitations can be overcome by adding new rules to the dependency grammar without having to make major changes to the parsing algorithm itself.

7 References

Covington, Michael A. 1990. A dependency parser for variable-word-order languages. Research Report AI-1990-01, Artificial Intelligence Programs, University of Georgia.

Covington, Michael A. 1994. GULP 3.1: An extension of prolog for unification-based grammar. Research Report AI-1994-06, Artificial Intelligence Programs, University of Georgia.

Covington, Michael A. 2003. A Free-Word-Order Dependency Parser in Prolog. Manuscript, University of Georgia.

Hudson, R. 1984. Word Grammar. Basil Blackwell.

Kwon, H.; Yoon, A. 1989. Unification-based dependency parsing of governor-final languages. Proceedings of the Second International Workshop on Parsing Technologies : pp.182-192.

Sohn, H. 1999. The Korean Language. Cambridge University Press.