Construction of a Knowledge Base by using Korean Text

Gi-Chul Yang and Key-Sun Choi

Dept. of Computer Science & Statistics
Mokpo National University
Mokpo, Korea

Dept. of Computer Science
KAIST, DaeJeon, Korea

Abstract

There are some difficulties in using natural languages as knowledge representation languages for computer systems. However, natural languages are the most frequently used knowledge representation languages for humanbeings. A way of automatic construction of knowledge-base by using Korean text is described in this paper. Dependency grammar has been used for parsing and the meaning of each sentence is represented with a conceptual graph.

1. Introduction

There are some difficulties in using natural languages as knowledge representation languages for computer systems. However, most human knowledge is encoded and communicated via natural language and a huge number of new textual documents become available on-line everyday. These creates the need for more sophisticated information and knowledge processing techniques based on natural language processing and knowledge representation techniques.

A mechanism of constructing knowledge-base using Korean text is described in this article. The knowledge-base constructor gets Korean sentences as its input and produce a knowledge-base. Each sentence in a text is converted into a conceptual graph before stored in the knowledge-base. Dependency grammar has been used for parsing and the meaning of each sentence is represented with a conceptual graph.

Korean is a partially free word-order language which has frequent elimination of necessary constituents and the roles component are decided not by their positions in a sentence but by their forms. Phrase structure grammar against dependency grammar is suit to the language such as English which has pretty fixed word-order in a sentence, but it has difficulties in parsing free word-order language such as Korea, Latin and Russian, since it requires many rules and complex processing. Dependency grammar works on binary relation between two syntactic units which represents governor-dependent relations. The groundwork of dependency grammar is started by Tesniere[6]. Hays[1], Robinson[4], Mel'cuk[2] and so on are contributed to develop its theory. It describes languages by using dependency relation between syntactic units and it has advantages such as simplicity and robustness over the syntactic rule based grammar. Hence, the dependency grammar is more efficient than the phrase structure grammar for Korean parsing.

Conceptual graph is generated from the result of the parsing. Conceptual graph is introduced in section two and Korean parsing with dependency grammar is described in section three. In section four, conceptual graph generation process is explained and organization of the knowledge-base is described in section five. The paper is concluded in section six.

This work was supported in part by the Korea Science & Engineering Foundation under contract 941-0900-030-2.

2. Conceptual Graph Basics

Conceptual graph is a graph based logic language which is more powerful than predicate calculus. Conceptual graph has graph notation like the idea of Peirce's existential graphs[3] to simplify the rules of logic and can represent higher-order relations and modalities. Graph notation has advantages in that diverse graph manipulation operations are available and easy to figure out the global status.

A conceptual graph is a finite, connected, bipartite graph. There are two kinds of nodes; concept nodes (displayed as a box in graph notation) which represent entities, attributes, states, and events, and relation nodes(displayed as a circle in graph notation) which represent the relationship among concept nodes. A single concept by itself(Fig. 1) may form a conceptual graph but it is not the case with the relation (Fig. 2), since every relation node should have one or more arcs each of which must be linked to some concepts(Fig. 3).

[CAT] (ANT)

(Fig.1) (Fig.2)

[EAT]->(AGNT)->[CAT]

(Fig.3)

A conceptual graph can be constructed by assembling percepts. In the process of assembly, concept relations specify the role that each percept plays and concepts represent percept themselves. A concept can be an individual or generic. The function referent maps concepts into a generic marker of a set I = { #1, #2, #3, ...} whose elements are individual markers. The function type maps concepts into a set of type labels. A concept with type(c) = t and referent(c) = r is displayed as (t) in the linear form.

Types of concepts (type labels) are organized into a hierarchy called type hierarchy. Type hierarchy forming operations includes conjunction and disjunction operations, so the type hierarchy will be a formal lattice. The type hierarchy constitutes a partial ordering and becomes a type lattice when all the intermediate types are introduced. The type lattice has the undefined type at the top and the absurd type $\perp$ at the bottom. There are lattice operations such as subtype $\leq$, maximal common subtype $\cap$, and minimal common supertype $\cup$. Amon arbitary type labels t1, t2, t3 if $t1 \leq t2$ and $t1 \leq t3$ then $\perp \leq t1 \leq t2 \leq t3$ or $\perp \leq t1 \leq t3 \leq t2 \leq$. Also $t1 \leq t2 \cap t3$. If $t1 \leq t3$ and $t2 \leq t3$ then $t3 \geq t1 \cup t2$.

Woods argue that it is necessary to represent intensional concepts which can not be represented in first-order logic, and conceptual graph can support statements about types and sets. Truth about statements concerning types such as CAT $\cup$ DOG (Carnivore) is established by intension, and statements about sets such as(δ represent denotation) δCAT $\cup$ DOG(set of all cats and dogs) by observing their extensions.

A conceptual graph has the translation operator φ which can translate a conceptual graph into first-order predicate calculus. For example, a sentence 'A monkey eats a walnut' can be represented in conceptual graph as in u:

u=[MONKEY]<-(AGNT)<-[EAT]->(OBJ)-> [WALNUT]

and

$$\varphi_u = \exists_x(\exists_y(\exists_z(\text{MONKEY}(x)\ \&\ \text{AGNT}(y,x)\ \&\ \text{EAT}(y)\ \&\ \text{OBJ}(y,z)\ \&\ \text{WALNUT}(z)))).$$

When φ is applied to u, φ generates constants and variables for the individual concepts and the generic concepts of u respectively. For examples, a conceptual graph

u' = [MONKEY] <- (AGNT) <- [EAT] -> (OBJ) -> [WALNUT:123]

is translated into

$$\varphi_{u'} = \exists_x(\exists_y(\exists_z(\text{MONKEY}(x)\ \&\ \text{AGNT}(y,x)\ \&\ \text{EAT}(y)\ \&\ \text{OBJ}(y,\#123)\ \&\ \text{WALNUT}(\#123)))).$$

There are two generic concepts ([MONKEY] and [EAT]) and an individual concept ([WALNUT:123]), thus two variables (x and y) and a constant (#123) is introduced by the operator φ. The individual marker on a concept must conform to its type label. The conformity relation :: relates type labels to individual markers, if t :: i is true, then i is said to conform to type t.

3. Parsing with Dependency Relations

Dependency Relation is defined as a binary relation between two syntactic units X and Y which represents governor-dependent relation. If Y depends on X, it is representable as $X \rightarrow Y$ and X, Y are called as governor and dependent respectively. Dependency Relation is a directed anti-symmetric, anti-reflexive, and anti-transitive. Sentence analysis with dependency grammar is the process of finding dependency structure of the sentence by using dependency relations among the syntactic units . Dependency structure organized as a tree. Syntactic categories assigned to each syntactic unit and dependency relations defined in between syntactic units are important factors in sentence analysis.

Dependency category is assigned to word-phrase to decide dependency relation between word-phrases. Two kinds of dependency categories are exist for the parser. One for word-phrases which are used as dependent and the other for word-phrases which are used as governor. They are called left-category and right-category respectively. If a word-phrase is used as a dependent the right-category of the word-phrase is considered for checking dependency relation. Left-category is considered if the word-phrase is used as a governor. There are 20 left-categories and 32 right-categories to define grammatical role of each word-phrase in a sentence. A part-of-speech which decide the grammatical role in a sentence is depends on functional word attached to the content word if the content word is an uninflected word. If a content word is inflected word, part-of-speech depends on the ending of the word. Therefore, the right-category can be divided as particle, ending of a word, modifier. Also, an uninflected word can be used as a word-phrase without auxiliary word.

Dependency Relation is defined in between left-categories and right-categories. There are dependency relations in between inflected words and noun, inflected word and adverb, and so on. If we depends only on the dependency relation between dependency categories to analyze a sentence, incorrect results can be produced in some cases. To reduce the incorrect result, restrictions on dependency relation should be used. The restrictions which most affected to Korean sentences are location restriction, order restriction, unique restriction and compulsory restriction.

Location restriction states that two word-phrases should be consecutive in a sentence to have dependency relation. Order restriction decides the word-order of each dependents which are depends on a governor word-phrase. Unique restriction tells that there exists only one dependency relation between certain dependency categories. Compulsory restriction indicates that word-phrase of right category should have a dependency relation to compulsory constituent.

The translation system is composed with morphological analysis module, syntactic analysis module and conceptual graph generation module. Syntactic analysis module constructs dependency structure by using the output of morphological analysis and information in dependency relation table which contains restrictions on dependency relations. The conceptual graph generation module produce conceptual graphs corresponding to each input sentences by applying converting rules to the dependency structures.

The process of morphological analysis and syntactic analysis are described by using an example sentence. A Korean sentence is composed with word-phrases. A word-phrase is a basic unit for separating words in a sentence. A word-phrase is composed with a content word which contains the meaning and a functional word which represents relation among words. Each word-phrase is composed with morphemes. A morpheme is a smallest unit of a language which has corresponding meaning to certain sound.

Following is the execution result of the parser for a sentence:

"vaticansie	itnun	arumdaun	sendangenun
Vatican	in	Beautiful	Church

Kyohwangyi	sanda."
Pope	Reside

The result of morphological analysis is:

```
vaticansie.
        vaticansi. 11 0 0
        e. 41 0 39
        vaticansi. 11 0 0
        e. 41 0 38
itnun.
        it. 61 38 14
        nun. 93 13 1
arumdaun.
        arumdau. 70 29 42
        n. 93 16 1
sengdangenun.
        sengdang. 12 0 22
        enun. 41 0 38
Kyohwangyi.
        Kyohwang. 11 0 0
        yi. 41 4 11
sanda.
        sa. 62 1 32
        nda. 91 12 0
        .. b1 1 1
        sa. 62 18 11
        nda. 91 12 0
        .. b1 1 1
        sa. 62 1 32
        n. 82 11 12
        da. 91 0 0
        .. b1 1 1
        sa. 62 18 11
        n. 82 11 12
        da. 91 0 0
        .. b1 1  1
```

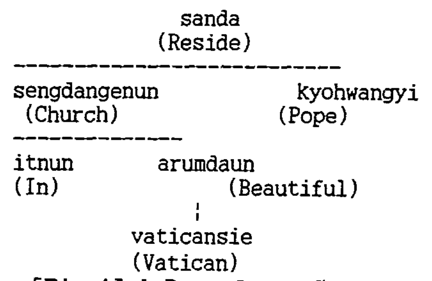
The number after each morpheme is a morphological information code. It is six-digit code(eg., 'ABCDEF') and contains information for morphological/syntactic analysis. The first two digit 'AB' indicates morpheme's part-of-speech and the first four digit 'ABCD' contains the left-categorial information. The right-categorial information is represented in 'ABEF'.

For example, the word-phrase 'Vaticansie' is consists of a content word 'Vaticansi' (i.e., Vatican City) and a function word 'e'. The function word 'e' can be used for both directional postposition and locative postposition. Therefore, the word-phrase 'Vaticansie' can be composed in two different way. The first one is 'Vaticansi' plus locative postposition 'e' and the second one is 'Vaticansi' plus directional postposition 'e'. It is shown in the result of morphological analysis in (1). The morphological information code '11 0 0' besides the word 'Vaticansi' indicates that the word's part-of-speech is proper noun. The code for the first 'e', '410039' tells that the part-of-speech of 'e' is adverbial particle ('41') and directional ('39'). The code for the second 'e' is '410038' that indicates the part-of-speech is advibial particle like the first 'e' and it is locative ('38'). The correct one, among those possible compositions, will be selected in the process of syntactic analysis.

The result of syntactic analysis is

```
vaticansie[11,41](loj,loc,0.890000)->
                                  itnun[61,93]
arumdaun[62,93](mdf,NON,0.890000)->
                          sengdangenun[12,41]
Kyewhangyi[11.41](sbj,agt,0.790000)->
                                 sanda.[61,b1]
itnun[61,93](mdf,NON,0.780000)->
                          sengdangenun[12,41]
sengdangenun[12,41](av4,NON,0.680000)->
                                  sanda[61,b1]
```

Above relations are the selected dependency relations among many possible dependency relations. The first relation, "Vaticansie[11,41] (loj,loc,0.890000)-> itnun[61,93]", tells that a word-phrase 'Vaticansie' has locational dependency to a word-phrase 'itnun'. Also, Vaticansie[11, 41] indicates the composition of proper noun (11) and adverbial particle (41) are selected for a word-phrase 'Vaticansie'. The information, (loj, loc, 0.890000), in between two word-phrases vaticansie[11,41] and itnun[61,93] tells that the relation is locative and the probability is 0.890000. Following final dependency structure can be drawn from the result of syntactic analysis.



[Fig.4] A Dependency Structure

Following conceptual graph is generated from the above dependency structure.

```
[RESIDE] -
  (AGNT) -> [POPE]
  (LOC) -> [CHURCH] -
         (ATTR) -> [BEAUTIFUL]
         (LOC) -> [CITY: Vatican].
```

4. Generation of Conceptual Graph

Conceptual graph generator receives the parsing result as its input. The parsing result is a dependency structure of input sentence. The dependency structure of a sentence "vaticansie itnun arumdaun sendangenun Kyohwangyi sanda" is shown in Fig 4. We start the conceptual graph building process from the root node in a dependency structure tree. The root node of our example tree is 'sanda'. Hence, the corresponding concept [RESIDE] is retrieved from the concept catalog. Next, the child nodes of the root node will be considered.

A conceptual graph is consist with concept node(s) and relation(s). Hence, the concept(s) and relation(s) should be extracted from the dependency structure. Those concepts and relations are stored in a dictionary (i.e., catalog). Dependency structure is traversed in depth-first order and the corresponding concepts/relations to each node are extracted from the catalog. These concepts and relations are joined together to build a conceptual graph.

After the concepts [RESIDE] and [CHURCH] are retrieved, they are joined together as [RESIDE] -> (LOC) -> [CHURCH]. The relation node (LOC) in between [RESIDE] and [CHURCH] is used since the dependency relation between two word-phrases 'sengdangenun' 9i.e., [CHURCH]) and 'sanda' (i.e., [RESIDE]) is av4. The dependency relation av4 indicates that the dependent word-phrase is the location of governor. The third step is taking the concept [POPE] for 'Kyohwang' and attached it to the conceptual graph. At this time, the relation (AGNT) will be used as the relation in between [RESIDE] and [POPE]. The result is

```
[RESIDE] -
          (LOC) -> [CHURCH]
          (AGNT) -> [POPE].
```

The next node in a dependency structure tree is 'itnun'. (LOC) is the corresponding conceptual relation in the catalog. The relation (LOC) is attached to [CHURCH] and produce

```
[RESIDE] -
          (LOC) -> [CHURCH] -> (LOC)
          (AGNT) -> [POPE].
```

After that, take the node 'arumdaun'. 'arumdaun' has (ATTR) -> [BEAUTIFUL] as an entry in the catalog. Also, 'arumdaun' is a child node of 'sengdangenun' [CHURCH]. Hence, it is attached to [CHURCH] and produce

```
[RESIDE] -
     (LOC) -> [CHURCH] -
          (LOC)
          (ATTR)-> [BEAUTIFUL]
          (AGNT) -> [POP].
```

Finally, the node 'vaticansie' is treated same as the previous nodes and produce a conceptual graph which is contains the same meaning of initial sentence "vaticansie itnun arumdaun sengdangenun kyohwangyi sanda". The final version of the conceptual graph is

```
[RESIDE] -
     (AGNT) -> [POP]
     (LOC) -> [CHURCH] -
          (LOC) -> [CITY: Vatican]
          ->(ATTR) -> [BEAUTIFUL].
```

The conceptual graph generation process can be accomplished by the following steps:

- Take the root node from the dependency structure.
- Find corresponding concept from the catalog.
- Take remaining nodes in depth-first order.
- Retrieve coresponding concept / relation from the catalog.
- Join currently retrieved concept / relation to the concept/relation of parent node in dependency structure.

5. Organization of the Knowledge-Base

The internal representation form of conceptual graph for the knowledge-base is U-Form[9]. U-Form start with the concept which has the most outgoing arcs (called head-concept). If there are more than one head-concept then choose the concept in alphabetical order for the first head concept.. After select the first head-concept, choose the relation on outgoing arc(s) of the chosen concept in alphabetical order. The syntax of U-Form is :

```
U-Form  :=  ((DummyRelation    head-concept
                          Subgraph* )+    )
Subgraph:= - relation  concept  Slist *
Slist   :=  relation  concept
DummyRelation :=  relation  |  h.
```

From the above notation, relation/concept is each relation/concept in a graph. The symbol 'h' is a dummy relation for the head-concept with no incomming arcs. For example, a sentence "monkey eat a walnut with a spoon made out of the walnuts shell." Can be represented with conceptual graph as shown below[5].

```
[EAT] -
 (AGNT) -> [MONKEY]
 (OBJ) -> [WALNUT] -> (PART) -> [SHELL:*x]
 (INST) -> [SPOON] -> (MATR) -> [SHELL:*x].
```

The corresponding U-Form is :

```
((h  EAT - AGNT   MONKEY
       - INST   SPOON  MATR   SHELL:*X
       - OBJ   WALNUT   PART   SHELL:*X)).
```

In this U-Form EAT is a head-concept. Threre are three '-'s which indicates that there are three outgoing arcs from the EAT node. Unlike other graph notation U-Form has only one form for any conceptual graph since there is a strict ordering and all the arcs are implicit in a conceptual graph.

The knwledge-base builder gets conceptual graphs represented with U-Form and separate each graph with (relation concept) pair. The concept in a pair placed under the flag with the relation. Hence, the name of the relation can works as a relation table name in a relational database. Here are three sentences to be used to show the knowledge-base structure. The sentences are:

a) John buy a car.

```
[PERSON:John]<-(AGNT)<-[BUY]->(OBJ)->[CAR].
```

b) Tom drive a truck fast.

```
[DRIVE] -
     (AGNT)->[PERSON:Tom]
     (OBJ)->[TRUCK]
     (MANR)->[FAST].
```

c) Nancy gave a book to John.

```
[GIVE] -
      (AGNT)->[PERSON:Nancy]
      (OBJ)->[BOOK:@1]
      (DEST)->[PERSON:John].
```

When the first sentence is comming into the knowledge-base the structure of the C-base will be

```
H    := {(1 BUY 3)}
AGNT := {(1 PERSON:john 3)}
OBJ  := {(1 CAR 3)}.
```

The first number is graph id (called G-id) and the number after concept indicates the length of the graph. Number of concepts in a graph will be the length of that graph. The length can be used to speed up the search by discarding the graphs which has different length in a exact match. After the second sentence has been accepted the C-Base became

```
H    := {(1 BUY 3)(2 DRIVE 4)}
AGNT := {(1 PERSON:john 3)(2 PERSON:Tom 4)}
OBJ  := {(1 CAR 3)(2 TRUCK 4)}
MANR := {(2 FAST 4)}.
```

A new relation MANR has been introduced to C-Base since there was no relation named MANR in the first sentence. The final C-Base looks like:

```
H    := {(1 BUY 3)(2 DRIVE 4)(3 GIVE 4)}
AGNT := {(1 PERSON:john 3)
        (2 PERSON:Tom 4)(3 PERSON:Nancy 4)}
OBJ  := {(1 CAR 3)(2 TRUCK 4)(3 BOOK:@1 4)}
MANR := {(2 FAST 4)}
DEST := {(3 PERSON:John 4)}.
```

The meaning match can be performed through this C-Base. For example, a query "Who drive a truck fast?" is translated to a conceptual graph

```
[DRIVE] -
      (AGNT)->[PERSON:?]
      (OBJ)->[TRUCK]
      (MANR)->[FAST].
```

Also, it is represented with a U-Form

```
((h DRIVE - AGNT PERSON:?
          - MANR FAST
          - OBJ TRUCK)).
```

This query graph is saperated into (relation concept) pair. There are four pairs in this case. Those are (h drive), (agnt person:?), (manr fast), (obj truck). Now, take a pair from the last one and search the C-Base. In this case, take a pair (obj truck) and look at the OBJ set and find (2 TRUCK 4). Next pair is (manr fast) and search the MANR set. (2 FAST 4) is found at this time. Each time any new elements are found then G-ids of those elements are intersected with the G-ids of old elements. Hence, the G-id '2' is remain so far. For the pair (agnt person:?), three elements (1 PERSON:john 3),(2 PERSON:Tom 4) (3 PERSON:Nancy 4) are found. After the intersection only G-id '2' is left. At last, for the pair (h drive) we get (2 drive 4) and get G-id '2'. All the found elements are used to reconstruct the original graph (graph for the second sentence). And get the answer PERSON:Tom.

Both exact matching and partial matching are possible. In the case of partial matching, both syntactic and semantic difference can be covered. Foe instance, we can get an answer with the both queries "Who drive a truck fast today?" and "Who drive a vehicle fast?" by using partial match with the current C-Base. In the case of semantic partial match, the concept type hierarchy is utilized.

6. Conclusion

The process of constructing a knowledge-base by using Korean text is described in this paper. The process requires Korean parsing, conceptual graph generation, and knowledge-base building. Dependency grammar has been used for Korean parsing because it is more efficient than phrase structure grammar for parsing free word-order language like Korean. The parser based on dependency relation has been implemented with C. Currently, the conceptual graph generator is being built. The knowledge-base constructor and matching processor has been implemented with CommonLISP. The integrated system will be implemented with C in the near future.

References

[1] Hays, D., Dependency Theory:A Formalism and Som Observations, Language, 40: 4, 511-525, 1964.

[2] Mel'cuk, Dependency Syntax:Theory and Practice, State Univ. of New York Press, 1988.

[3] Roberts, D., The Existential Graphs of charles S.Peirce, Mouton, The Hague, 1973.

[4] Robinson, J., Dependency Structures and Transformational Rules, Language, 46: 2, 259-285, 1970.

[5] Sowa, J., Conceptual Structure: Information Procesing in Mind and Machine, Addison Wesley, Massachusetts, 1984.

[6] Tesniere, Claude Aux fondements de la syntaxe:I'ergatif, Paris, Press Universitaires de France, 1959.

[7] Yang, G.,Y. Choi, & J. Oh, CGMA: A Novel Conceptual Graph Matching Algorithm,7th Internaational Workshop on Conceptual Graphs, Las Crusis, 1992.