Automatic Relation Triple Extraction by Dependency Parse Tree Traversing

DongHyun Choi and Key-Sun Choi

Computer Science Department
Semantic Web Research Center, KAIST
Daejeon, Korea
`cdh4696@world.kaist.ac.kr,kschoi@cs.kaist.ac.kr`

Abstract. To use the information on the web pages effectively, one of the methods is to annotate them to meet with ontology. This paper focuses on the technology of extracting relation triplets automatically by traversing dependency parse tree of a sentence in postorder manner, to build ontology from plain texts.

1 Introduction

Problem that prevents ontology from widespread using is that it is hard to build. To build ontology automatically, we need to acquire relation triplets from text automatically. Relation triplet acquisition can be divided into two procedures - relation triplet extraction and mapping triplets into one of predefined relations. In this paper, we propose a method to extract relation triplets from text, by traversing dependency parse tree using predefined rule sets.

Table 1. Relation extraction result of a sentence: "James visits a company which has held seminar in London."

Sentence: *James visits a company which has held seminar in London.*
Result:
Triple1: (*James*, *visit*, *company* AND (Process *holding* AND (Objective *Seminar*) AND (in *London*)))

2 Relation Extraction System

2.1 Overview

The overall architecture of this system is as follows: after parsing the given sentence using dependency grammar, seven preprocessing procedures are executed to give more information on decision tree for rule application to extract the relation triplets. After preprocessing, we traverse the resulting dependency tree in postorder to find the relation triplets by using predefined generic hand-written rule sets. In order to solve the long-distance problem, we need to transmit the information at the lower part of dependency tree to the upper part of dependency tree. To do that, we use RT(Reserved Term), RC(Reserved Clue) and RQ(Relation Queue). RT contains a single term which will be used as concept. RC contains a 'clue', which will be used to determine the kind of relation. RQ contains set of relation triplets which are extracted so far.

2.2 Preprocessing module

Term Marking In this phase, we mark terms in the dependency tree. Terms will be used as concept/instance in the resulting ontology.

Named Entity Recognition This phase assign the words to semantic information. Semantic information can be used to map the extracted triplets into the predefined set of relations.

Marking To-infinitive/Gerund Since To-infinitive/Gerund is a verb which is used as object of the other verb, we need to consider them differently from the other verbs.

Processing Coordinate Conjunction Coordinate conjunctions connecting verbs show two different cases in its dependency structure: (1)Two verbs share some contents(ex. subject), (2)Two verbs do not share any contents. For each case, we should change the parse tree so that we do not need to consider about coordinate conjunctions seperately.

Relative Anaphora Resolution Relative anaphora like 'which' or 'who' refers to another term in the sentence. Since we need to make relation with the term which is refered, not with 'which' itself, we need to resolve the relative anaphora.

Marking Action Action is a concept of ontology to be built, which represents action or status change of some object. We use Actions to gather two or more relation triplets which should be represented as a composite one. In table 2, result (1) does not mean that Samsung holds seminar in London - rather, it gives two partial informations which are wrong if they are not gathered. Thus, we use the concept of Action to get the triplet of result (2).

Merging Negation/Frequency with Verb Negation/Frequency information is merged with its attributed verb. Considering the sentence "James is not a student", we mark 'not' at the node 'is' so that the extracted relation triplet does not become (*James*, *ISA*, *student*).

Table 2. Relation Extraction from a sentence "Samsung has held seminar in London."

Sentence: *Samsung has held seminar in London.*
Without_Action – Result (1):
Triple1: (*Samsung*, *have hold*, *seminar*)
Triple2: (*Samsung*, *have hold in*, *London*)
With_Action – Result (2):
Triple3: (*Samsung*, *Process*, *Holding* AND (Objective *seminar*) AND (in *London*))

2.3 Dependency Tree Traversing Module

In this module, we extract relation triplets by post-order dependency tree traversing. Figure 1 shows the procedure of extracting relation triplets of sentence, "James visits a company which has held seminar in London", and the extracted triplets.

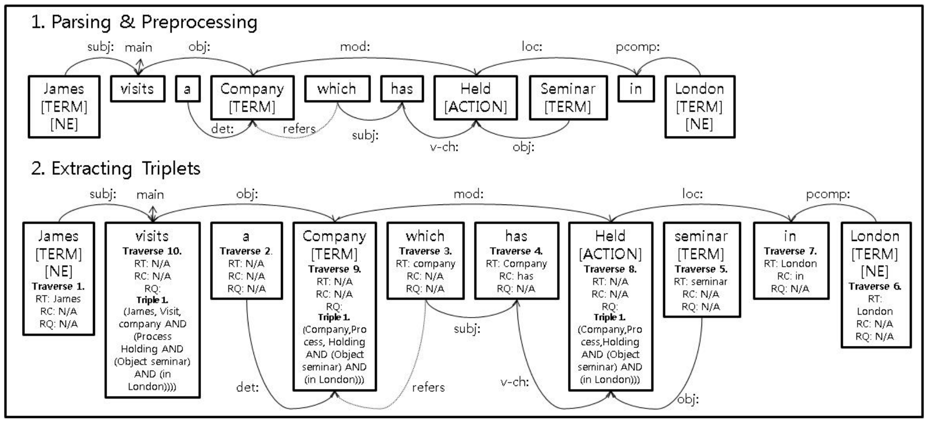
1. Parsing & Preprocessing
subj:
main
obj:
mod:
loc:
pcomp:
James [TERM] [NE]
visits
a
Company [TERM]
which
has
Held [ACTION]
Seminar [TERM]
in
London [TERM] [NE]
det:
refers
subj:
v-ch:
obj:
2. Extracting Triplets
subj:
main
obj:
mod:
loc:
pcomp:
James [TERM] [NE] Traverse 1. RT: James RC: N/A RQ: N/A
visits Traverse 10. RT: N/A RC: N/A RQ: Triple 1. (James, Visit, company AND (Process Holding AND (Object seminar) AND (in London))))
a Traverse 2. RT: N/A RC: N/A RQ: N/A
Company [TERM] Traverse 9. RT: N/A RC: N/A RQ: Triple 1. (Company,Process, Holding AND (Object seminar) AND (in London)))
which Traverse 3. RT: company RC: N/A RQ: N/A
has Traverse 4. RT: Company RC: has RQ: N/A
Held [ACTION] Traverse 8. RT: N/A RC: N/A RQ: Triple 1. (Company,Process,Holding AND (Object seminar) AND (in London)))
seminar [TERM] Traverse 5. RT: seminar RC: N/A RQ: N/A
in Traverse 7. RT: London RC: in RQ: N/A
London [TERM] [NE] Traverse 6. RT: London RC: N/A RQ: N/A
det:
refers
subj:
v-ch:
obj:

Fig. 1. Relation extraction example - step by step

3 Conclusion

This algorithm gives solution to long-distance problem, which cannot be solved using pattern matching method. Also, this algorithm extracts not only relation triplets but also the constraints of the arguments of triplets. This will surely enhance the quality of ontology built using the resultant triplets of this algorithm.

Acknowledgments. This work was supported in part by MKE & IITA through IT Leading R&D Support Project.

References

1. Hearst, M.A.: Automatic Acquisition of Hyponyms from Large Text Corpora. In: Proceedings of the 14th conference on Computational linguistics, pp.539-545. Association for Computational Linguistics, Nantes (1992)
2. Ravichandran, D., Hovy, E.: Learning Surface Text Patterns for a Question Answering System. In: Proceedings of the 40th Annual Meeting on Association for Computational Linguistics, pp.41-47. Association for Computational Linguistics, Pennsylvania (2001)