A Syntax Parser for Korean Language

Project code

Taher ROMDHANE

I Introduction

I.1 Goal

The project tends to implement a grammar for a new language other than English or French, which has been already demonstrated in the lab courses during the study. As a language learner of Korean (which is a fascinating language by the way, and easier to learn than it seems), which is largely different with any western language, we consider it interesting to apply the syntax analysis to Korean.

Since Prolog doesn't have much support for Korean characters (we have to resort to Romanization, i.e. writing Korean with latin characters which isn't authentic), we decide to realize the application on Python, which support almost any character encoded in "utf-8". The environment we used is based on "python3.6+" and it is based on KoNLPy, a Python package for natural language processing (NLP) of the Korean language. Please follow instructions in the github repo for installation and trouble-shooting. You can find the code here.

I.2 Challenge for the Syntax Tree Implementation

While having a simple alphabet, Korean is a very rich language grammatically and has a unique writing system, thus, the parsing of texts of this language has many unique problems to solve. The first challenge is that Korean is written as blocks of syllables, that form words. But most importantly, the grammar is very complex, this includes many different grammatical particles that should be taken into account, that sometimes even change shape depending on the preceding sound. For example, -eun/-neun (-은/-는) and -i/-ga (-이/-가) . Sometimes sounds may be inserted instead. Not to mention also the three levels of speech, which are used to show different levels of respect. In the end, there are preferences and restrictions on how words should be placed and used. Thus, new grammar rules are required for the syntax analysis. The grammar we implemented is a simple one at yet should be enough for generalized usage.

II Word Segmentation

II.1 Parsing Algorithm

The algorithm used for the word segmentation is an ad-hoc, recursive-descent algorithm given a Korean sentence POS-list. The algorithm takes in the whole sentence and tries to split it gradually into the constituting elements, in a recursive manner. For example, a sentence is split into a subordinate clause + a main clause. The main clause is split into a phrase + a predicate, and the subordinate clause into a phrase + verb-phrase + connecting suffix etc… Until it finds the constituting atoms of the sentence.

The algorithm works as follows :

- Pass the input string throught the KoNLPy morhpeme analyzer to get a list of the individual parts-of-speech it contains
- Apply a series of mappings to that morpheme list that distinguish and provide more detail about specific morphemes and morhpeme patterns
- Pass the mapped morpheme list through one of two available phrase parsers to get a nested parse-tree of phrase structures in the input
- Apply an annotation pass to the parse-tree, adding descriptive detail, definitions, references, re-labelings, and so on for a more useful display

II.2 Limitations

The exact implementation can be found in the source code, in the “rd_grammar.py” file. I think the algorithm performs well enough for a general purpose since we have many test sentences and does a good enough job to parse them. It might have its limitations however especially with long sentences since they become more and more complicated.

III Grammar Implementation

III.1 Basic Grammar

As an SOV (subject-object-verb) language, a Korean sentence can still be separated to noun phrases and verb phrases. However, this is a bit more complex than SVO languages since the verb

must be at the end of the sentence. It still has a clear pattern for sentences though.

For noun phrases, all modifiers, including adjectives, numbers and particles should be placed in front of the noun. Similarly, adverbs should also generally appear before the verb.

III.2 Particles

In the Korean language, particles are a huge part of it since they denote everything from topic (은/는 (eun/neun)), subject (이/가 (i/ga)), object (을/를 (eul/reul)), number (들 (deul) for plural), time and location (에/에서 (e/e-seo)) etc… They also differ with respect to the level of speech. Especially in the written language (because spoken language might omit some of them in informal speech), they make our life easier when it comes to parsing a sentence, since they can give a huge indication about what a word's role is and its relation to the other words.

III.3 Limitations

The grammar of the Korean language is rather complex and the vocabulary is rich. This might prove to be a limitation for the algorithm as sentences can be very complicated.

Many words can be regarded as different types of words, for example either a noun or a particle, which might prove confusing for the algorithm. Listing all possible rules in the grammar is tedious and may generate a wrong parse tree.

IV Conclusion

This implementation was interesting to make as an avid learner of the Korean language. In the end, it can pretty much handle in general most of the simple sentences, with some limitations for longer ones.

In terms of further improvement, maybe I can explore the more complex structures that Korean language has to offer, as well as a more rich vocabulary and the more obscure grammar structures. Moreover, maybe try to adapt it to a more informal setting (like social media for example) so that it keeps up with modern uses for such an algorithm.