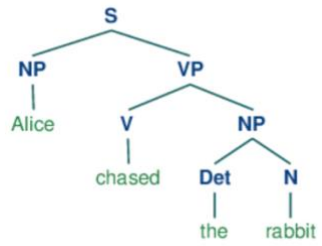
20170396 예성현

The implementation is divided into three parts; chunking, extracting output, evaluating the output. The five target verbs are ‘activate, inhibit, bind, induce, abolish’. For high performance, POS tag result from constituency parser in allennlp is used. Grammar is composed of six chunks: PARA (to exclude terms inside parenthesis), NP, PP, VP, CLAUSE, and EQUAL (to detect apposition clause surrounded by commas). The grammar is based on the one from Chapter 7-4 in the textbook using regexparser, therefore resulting in a nested structure.

After chunking, result string is extracted by traversal of tree chunk through recursion. For each child of a node, if a NP chunk is encountered, NP chunk, its parent, and the depth of the node is stored in a tuple. If a PP chunk is encountered, PP chunk starting with ‘by’ is treated specially. If the parent chunk has the target verb with another verb coming before (is, was, were…), it is interpreted as passive clause. Therefore, the output form should be like ‘A was inhibited by B’. If the node is an individual tuple rather than a chunk, the module first checks if it is one of the five target verbs and also if the tag is a verb. Then for elements in the list where NP chunks are stored, only chunks that has 1 less depth than the verb should be picked. Also, the parent of VP chunk should be included in parent of NP chunk. This condition is easily grasped when thinking of the relation between ‘Alice’ and ‘chased’ in the tree below which is from the textbook.



From the NP chunk, only noun and number (tag with CD) is added on the result list. To extract ‘A and B activate C’ into ‘A and B, activate, C’, the module checks if two nouns are connected by CC tag word. After dealing with first and second entry, the flag bit is set so that the next NP chunk is now added to the third entry. Some results might return empty string if the POS tagger assigns wrong tags. For example, there are some cases that the tagger tags target verb words with noun. To complement the performance of the POS tagger, a backoff tagger is implemented which manually tags ‘VB’ to target verb word when the result string is empty. After backoff tagging, it goes through the extracting process again with a new tag. To handle multiple target verbs in a sentence (i.e. ‘A bind B’, ‘C inhibit D’), three words are grouped as a string.

Evaluation is done as the last step. For each element of the result list, which is a string consisted of three words, is compared with the answer data. For string comparison, the two string has to be exactly the same. If the result is correct, true positive is incremented, if wrong result came out, false positive is incremented, and if correct answer is missing, false negative is incremented. Using these values, Precision/Recall/F-score is calculated.

The performance of training data is Precision: 0.792, Recall: 0.773, F score: 0.783. Among 80 train sentences, 17 sentences have wrong output. Among wrong outputs, 7 can be interpreted as correct if relaxed criterion is used. For example, from the sentence “Beer and wine induce gastroesophageal reflux, mainly in the first hour after intake.”, the expected output is “Beer and wine, induce, reflux”. Note that the word ‘gastroesophageal’ is not included because it is an adjective. However, the allennlp tagger interprets the word as a noun and add it on the output. The output “Beer and wine, induce, gastroesophageal reflux” still conveys the correct meaning. Therefore, sentences like this could be interpreted correct if relaxed criterion is used. Another 5 wrong sentences also had some POS tagging issue but these could not be interpreted as correct even though relaxed criterion is used. These are the cases where the sentence includes multiple target verbs. For example, for sentence “Several of these analogs can induce SA-mediated defense and inhibit growth.”, the expected output is “analogs,

induce, defense", "analogs, inhibit, growth". However, the actual output is "analogs, induce, defense and inhibit growth" because 'inhibit' is interpreted as a noun. For a single target verb existing sentence, a backoff tagger solves wrong-tagged verb by manually tagging it to verb. However, for a multiple target verb existing sentence, the result is not empty but a wrong output instead, therefore not satisfying the condition for the execution of backoff tagger.

Remaining 5 sentences have grammatical issues. First major issue is distinguishing participle and verb. I already handled case for present and past participle by making '(VBG or VBN) + NN' as a NP chunk. However, POS tagger might wrongly tag the participles, leading to wrong result. For example, from the sentence 'However, recent research suggests that prolonged sedentary behavior might abolish these healthy metabolic benefits', 'prolonged' is tagged as VBD. Native English speakers would easily guess that 'prolonged' is used as participle because it is included in a 'that clause' and clause is mainly consisted of NP and VP. Because it is hard to directly distinguish past participle and a past tensed verb, we can solve this problem indirectly by using other grammar features such as 'that clause' for this example.

Another grammatical issue is distinguishing preposition and subordinating conjunction. Both are POS tagged as 'IN' in NLTK. However, they should behave differently when chunking. For example, if a sentence starts with 'In the 1990s, ~~', the phrase before the comma should be chunked as a PP chunk because it is consisted of <IN><NP>. If a sentence starts with 'Since PS-ASOs bind to major components, ~~', 'Since PS-ASOs' is chunked to PP even though the right chunking is 'Since, (CLAUSE (NP PS-ASOs) (VP bind (PP to major components))), ~~'. This is because firstly, POS tagger cannot distinguish preposition and subordinating conjunction and secondly, PP chunking is done earlier than CLAUSE chunking. Because it is hard to fix the tagger, we should focus on the parser. Because regexparser is a bottom-up parser just like shift-reduce parsing, small PP chunking is done earlier than large CLAUSE chunking. Therefore, one solution is to change the parser to left-corner parsers so that top-down approach is partially possible or to lookahead LR parser to complement the shortcomings of bottom-up parser.

For test sentences, the performance is Precision: 0.8, Recall: 0.8, F score: 0.8. All the wrong sentences are cases that are correct when relaxed criterion is applied, which was explained previously. It is notable that performance of training and test data is similar, which somewhat verifies the generality of the grammar.