[Overview]

The **key steps** of this problem can be stated like this. First, find some sentences which will work for training my own grammar and pre-process them. Then, chunk the 'essential' parts of the sentences so that it makes easy to extract those triples. Finally, traverse the chunked tree in appropriate manner and extract only essential parts, making correct triples.

Here, making a grammar for chunking was the most experimental part of this project. Since there was a limitation of the form of input verbs, I could categorize them in three categories. First, 'A action B' or 'B is actioned by A' triples were possible. Here, in addition to the NP chunking, I modified the grammar in order to extract verbs with modifiers, for example, 'possibly inhibit'. Second, 'A, that/which action B' can be extracted into <A, action, B> triples. Finally, 'A verb sth and action B' can also be extracted into <A, action, B>.

[Detailed Approach]

First was to find some sentences, preprocess and annotate them with expected triples. Here, preprocessing was done by word_tokenizing, pos_tagging and re-tagging some ambiguous words. While tagging some sentences, I could realize that various biological or chemical noun phrases are tagged with different POS rather than noun, so I manually annotated some of the ambiguous words into 'NN' and for some words which are not appearing in nltk wordnet, they are also tagged into 'NN' since the probability of tagging with wrong POS (for example, 'JJ') was too high. Also, in this part, for easier parsing and chunking, the phrases inside '(' and ')' were ignored, too. 'I' in Type I was also regarded as 'NN' since it is used as the number one, not the word meaning 'me'. Also, expected triples were annotated for each sentence, too.

Then, I made a grammar which can chunk and extract only some essential parts of the sentences. While defining a grammar, I made an assumption which I simply explained above. I categorized the structure of the sentences into three categories. First one was the simplest one, 'A action B' or 'A is actioned by B'. Here, Noun Phrases were chunked using various grammar rules which can cover the input sentences. The verb phrases with target verbs were also chunked into 'TVP'. Not only action, actions, actioned were included but also modifiers such as 'potentially', 'significantly' and 'can' were included in order to extract a big chunk of verb phrases. Second one was 'A, that/which action B'. First, 'that/which action B' phrase was chunked into 'THATT' in the code. Then, if there is a Noun phrase before THATT phrase, the noun phrase was expected to include 'A'. The third one was 'A verb sth and action B'. To chunk this structure, I had to chunk verb phrases even if they are not containing the target verbs. In this case, even though there is another phrase between the noun and the target verb phrase, A is thought to do the action of the target verb phrase. Here, 'and action B' was chunked into ANDT chunk, and then the whole sentence, 'A verb (sth) and action B' was chunked into ANDTARGET chunk.

Finally, I had to collect appropriate triples from chunked sentences. Here, I did this using nltk ParentedTree in order to traverse the tree easily. Since the first category is simple, it could be done by simply find A and B right next to the target verb phrase. However, to make each noun phrase simpler, I simply discarded the phrases after 'of', 'in', 'with', and 'as'. By this, I could get the key part of the extracted phrases. The second category was 'A, that/which action B'. Here, the very first noun phrase part was expected to work as 'A' in the triple, so it was extracted and simplified. THATT chunk, which is 'that/which action B' was found where TVP chunk was used to make 'action' part and the NP chunk following TVP chunk was used to make 'B' part of the triple.

Lastly, the third category, was 'A verb sth and action B'(ANDTARGET in the code grammar). Here, ANDTARGET consists of a noun phrase which works as 'A', and the leftover ANDT chunk which works as the whole leftover. So, first find the noun phrase and simplify as the first category. Then, look into the ANDT chunk, which consists of {<CC>?<TVP><NP.?>}, and find the TVP phrase. Simplified TVP will be the 'action' part in the output triples, and the simplified NP becomes the 'B' part of the triple.

Finally, after several rounds of editing the grammar in order to better extract from the training sets, Precision, Recall and F score were measured in validation set.

[Limitations and Improvements]

One of the biggest limitation in this approach is that pos tagging is a really erroneous step. Since there

are so many biological terms are used, it was hard to appropriately annotate the words with their POS. Here, I tried to re-tag some words, but some errors remained. For example, the phrase 'locally synthesized dihydrotestosterones' were tagged into 'RB', 'VBN', 'NN', thus they were not chunked into a noun phrase even if the 'synthesized' part was used to edit the word 'dihydrotestosterones', not in the verb way. By this reason, this sentence didn't give any triples. Also, tokenizing words were erroneous. Since there are various chemical terms used including '-', '(', and ')', those chemical terms were not tokenized into a single word.

In the step of making the grammar and chunking the words, it seems that I need a better grammar in order to find out the 'context' of the sentence. For example, if the sentence includes 'A activates B, none activates C, and D activates E', then my system extracts <none, activates, C> into one of the triples. Also, even if there are negative modifiers, it will still extract triples. Thus, I need a way to find the context and find if it is really doing the action or not. Also, some phrases like 'action not only A but also B' means that the action is on A and B, so the phrases like that have to be chunked into some special phrases which indicates the context.

In the tree parsing step, simplifying the phrases made some mistakes. The biggest mistake was occurred when discarding the words after 'of'. This made some 'variety of ~' chunk into 'variety'. Since this doesn't make any sense, there have to be a way to check whether the leading phrase or following phrase of the token 'of' is important in that specific phrase.

Here, I could got only one triple extracted from the sentences. Also, the code worked pretty bad for these cases so that they make a mixed triple, not even a single triple. By improving implementation details, it will make multiple proper triples.

Overall, there were various good-looking triples extracted. However, since the 'exact match' of the triples, including the whitespace and the scope of the phrase (for example, how many do I have to include the details of the noun phrase) made lower performance than real.

(for training set)
precision: 0.7
recall: 0.3111111111111111
F-score:0.43076923076923074

(for validation set)
precision: 1.0
recall: 0.6
F-score:0.7499999999999999