1. Method for addressing goal

The goal of this assignment is convert (sentence) into list of (triple). My methodology handles this problem by following:

(Raw sentence) ==[nltk word tokenizer]==> (Tokenized sentence) ==[nltk pos tagger]==>

(POS-tagged sentence) ==[preprocessing]==> (pre-processed tagged sentence) ==[CFG parsing]==>

(parsed tree) ==[relation extraction]==> triples

I used nltk.word_tokenize() and nltk.pos_tag() function for two precedent processes. The main stage is CFG parsing and relation extraction. Preprocessing was for fine-graining pos tag to feed to grammar parser.

But first, I would like to mention why I skipped NER process. First, nltk.ne_chunk() could not recognize named entities well, especially for biomedical terminologies. Secondly, checking the result without NER, I found that detecting named entity is not the key problem-if my extractor succeed to find a relation, the precision is quite decent-.

In the **preprocessing**, I added a number of things to do for compensating the imprecise tagger. There is a tendency that some verbs in 3rd single form are tagged to plural nouns, so I manually changed the tag especially for actions 'inhibits', 'prevents','binds','induces'. Also I regarded be p.p. by phrase into one verb and 'bind to' into one verb since it is phrasal verb. Finally I changed the overall pos tag make it suitable for my grammar. For example, aggregating 'VBP/Z/D' into 'VB'.

For **CFG parsing**, my grammar is :

```
CompleteS -> S '.'
S -> S 'conj' S | S ',' S | S ',' 'conj' S
NP -> NP 'conj' NP | NP ',' NP | NP ',' 'conj' NP
VP -> VP 'conj' VP | VP ',' VP | VP ',' 'conj' VP
AP -> AP 'conj' AP | AP ',' AP | AP ',' 'conj' AP
PP -> PP 'conj' PP | PP ',' PP | PP ',' 'conj' PP
S -> NP VP | Aux NP VP | 'WH' NP VP   | PP ',' S   |   S ',' PP | 'IN' S ',' S | AP ',' S | S ',' RelP | PastPartP ',' S | S ',' PastPartP | GP ',' S | S ',' GP
Sbar -> 'That' S
Aux -> 'MD' | 'DO_Aux'
NP -> 'DT' Nom | Nom | GP | ToinfP | NP ',' NP ','
Nom -> Nom PP | Nom RelP | Nom Sbar | Nom ',' RelP | Nom GP | Nom PastPartP | Nom 'CD' | 'CD' Nom | Nom ToinfP | Nom 'Bywhich' S |   AP Nom | Nom AP |
Nom 'NN' | 'NN'
VP ->   Aux VP | 'Beppby' NP | Vom NP | Vom PP | Vom PastPartP | Vom S | Vom Sbar | Vom
Vom -> 'VB' 'conj' 'VB' | 'VB'
AP -> 'JJ'
VP -> 'beVB' NP | 'beVB' PP | 'beVB' PastPartP | 'beVB' S | 'beVB' Sbar | 'beVB' AP | 'beVB'
PP -> 'IN' NP | 'TO' NP | 'IN' PastPartP
RelP -> 'WH' VP | 'That' VP
GP -> 'VBG' NP | 'VBG' PP | 'VBG' S | 'VBG' Sbar| 'VBG'
PastPartP -> 'VBN' NP | 'VBN' PP | 'VBN' S | 'VBN' Sbar| 'VBN'
ToinfP -> 'TO' 'VB' NP | 'TO' 'VB' PP | 'TO' 'VB' S | 'TO' 'VB' Sbar | 'TO' 'VB'
```

It's quite complex and not artistic. However, there is a reason why I let my grammar be not concise. At the beginning, I tried to build non-ambiguous grammar. But the goal of this homework is just extracting relations. In my grammar, even if ambiguity arises and many parsed trees are retrieved, the resulting relations did not differ much-except for cases where coordinate rule is cared a lot, I'll talk about it later-. So I sacrificed conciseness to add up the coverage of grammar, adding up new rules to cover training sentence.

Though there are some rules. First, I added coordinate rules for S, NP, VP. AP, PP. For NP, I first separated modifiers to 'complement' and 'adjunct'. Adjunct can be added to Nom(nominal) (ideally at the tail), recursively but complement make nominal into NP. Adjunct will be added ideally at the tail of nominal but I regarded that adjective and nominal can also be just nominal. GP, RelP, PastPartP, ToinfP is phrase starting with gerund, relational pronoun, past participle, to-infinitives , respectively. And rest are heuristically-added rules from train set like gerund phrase preceding sentence with comma…etc.

I didn't use featured grammar because I thought it is not much effective in this case. 'SUBCAT' feature, we don't know the subcategory from pos tagger. TENSE or AGR or NUM feature didn't seem to help a lot. So I decided just to stick with the simple CFG.

Finally the **Relation Extraction**. Now I have parsed tree with CFG, and possible five actions. For each action, find VP containing it. If exist, we need to perform two tasks 1) find subject noun 2) find object noun. For 2), find NP inside VP and find the very first noun traversed in that NP(because we only need to find one-word). For 1), there are two cases – (1) VP is inside RelP(Phrase of which header is relative pronoun so modifying noun) (2) VP is inside S(common case, S-> NP VP). First find the nearest parent that is either RelP or S. If RelP(case (1)) find NP that the RelP modifies and find noun from NP. Else if S, find NP in that S and find noun from NP. Now we retrieved <noun,action,noun> triple. But there can be too many triples. In this case, I got a heuristic rule. 'Choose the triple with shortest noun'. It is because, the case when different triples resulted is mostly caused by the coordinate rule. But I suggest it is inevitable. For the cases where, say, 'and' is used, it is 'possible' to read in many ways. Although, what I observed is that if 'clear' and 'make-sense' phrases that is connected with conjunction exist, the shortest noun most likely be that phrase(containing conjunction), not single noun without conjunction. This improved the result quality drastically.

2. Discuss result & Suggest Improvement

For the test cases, I got precision: 0.8333333333333334 recall: 0.5 f-score: 0.625. It is surprisingly high (at least in my POV). Moreover the f-score for test case does not vary a lot from it of training case. But I really want to say I did not see the test set, and I've guessed some reasons explaining the high score. First, because of relaxed-evaluation. What I heard from TA is if my result contains only one noun in the correct NE, that is considered to be correct. Secondly, the way I built the grammar. When I wrote the grammar with testing the training set, I usually considered general cases containing the current problematic cases. And also I didn't trained from the entire training set-I think this is somewhat important reason. Finally I guess I was very lucky.

Compared to surprisingly high precision, recall is kind of low. I assume the major reason is poor POS tagger. It show especially bad performance in tagging biomedical word and homographs(induced as past tense and adjective or past participle tense). So I have thought of using better tagger like AllenNLP but AllenNLP itself already parses the sentence so I think it doesn't really match the purpose of this relation extraction. Maybe it would be helpful if there is a dictionary for biomedical terms.

One problem occurred by that I removed adverb when tagging the sentence. I thought that adverb does not affect a lot in the relation extraction for this assignment. However, it made the parsing process difficult and forced me to add up some complex rules for handling those removed-adverb cases. It would be better to consider adverb, too.

For the 'triple with the shortest noun' heuristic for choosing the relation in ambiguous case, it has a limit that only one triple for one action can be retrieved(also, the correctness is not guaranteed for every case). I suggest using PCFG with the aid of large training set, in order to choose 'most likely' parsing form all 'possible' parsed trees caused by coordinate rule.

Finally, dependency grammar could be better for finding subject and object of the verb.