Methods

The whole process of this assignment can be divided into three steps: gathering the corpus data, designing the triple extractor, and evaluating the performance. Although the assignment description stated that the gathering of the sentences that contain relevant verbs (activate, inhibit etc.) does not have to be automated, I decided to semi-automate the task. I wrote a crawler script that checks all articles of PUBMED-MEDLINE for relevant sentences in the abstract. The advanced search option of the website helped me to designate the year to search in. I searched from year 2020 for the relevant sentences until 300 sentences were found. However, having collected 300 sentences does not mean that I had at least 20 sentences for each type of action verbs (positive, negative etc.). So I ran the script from where I left off until I had 20 sentences for each type of verbs. After collecting the sentences, I manually tagged the triples of each sentence because I thought it would be faster to do so instead of writing more code.

To design the extractor, I read a few sentences from the tagged corpus. There were some patterns and rules that could help me define the grammar to extract the triples. A triple contains a noun phrase X, an action verb phrase, and another noun phrase Y. I first extracted noun phrases from a sentence as a tree form. Some words are bound to be excluded from the noun phrases. I checked those words if they can be structured into a passive verb form. Some words might still not belong to any phrase. I checked those if they are active verb forms and structured them if necessary. Words that are left after this step are irrelevant ones such as prepositional phrases, and were ignored. A noun phrase and a verb phrase form a clause, and a clause is directly related to finding the triples. So I checked the clauses of a sentence and extracted triples from them.

After designing a triple extractor, I checked its performance using the testing code. I ensured the corpus contained 20 sentences for each of the verb types and split in 8:2 ratio, for training and testing, respectively. I applied the extractor to each of the test cases, and compared the result with the answer. By counting the true-positive, false-positive, and false-negative, the precision, recall and the f-score were calculated. A triple existing in a test case was the positive class, no triple existing in a test case was the negative class. Thus, I included additional 20 test cases in which sentences does not contain any triples to use as the negative class.

Discussion

The grammar rules are designed based mainly on the characteristics of English writing style, patterns visible in the corpus, plus some intuition. I first designed the rules so that it contains all rules (for noun phrases, verb phrases and clauses) in one string. However, such method limited the flexibility of prioritizing which to group first. So, I separated the grammars into noun, passive verb forms and active verb forms. This way I could parse noun phrases first, then passive verbs, then active verbs, then clauses. This prevents error cases in which a part of a noun phrase is stolen by a verb phrase that succeeds it.

There are six cases for noun phrase grammar:

```
NP: {<DT>?<JJ.?>*<NN.?>+}
NP: {<PRP.?><NN.?>+}
NP: {<NP><,>?<CC><NP>}
NP: {<NP><,><NP>}
NP: {<CD><NP>}
NP: {<NP><IN><NP>}
```

They cover most of the grammatical structures of noun phrases. The first one is the most basic case, “one or more of nouns, preceded by an optional article or an adjective”. But sometimes a pronoun preceded a noun, in cases such as “our findings require…”. This is covered in the second rule. Nouns are often listed together in cases such as “JP5 and BG1”, with an optional comma

before the ‘and’. This case is covered in the third rule. When more than 2 nouns are listed, they are chained with commas, as defined in the fourth rule. Since the corpus is a biological text, nouns are often preceded with numbers, i.e. “three binding sites” and “60nM”. Thus in the fifth rule, a cardinal digit precedes a noun phrase. The last rule covers cases in which a noun phrase contains prepositions such as ‘of’ and ‘with’. In the ideal circumstances, these rules should work flawlessly. However, relying on POS tags is not the best method, as it will be addressed later in the report.

The rules for verb phrases are divided into two cases: passive and active. Passive forms are defined as the following grammar: `PSV: {<VBD><RB.?>*<VBN><RB.?>*<IN>}`. It reads “a past participle form of verb preceded by a past tense of a verb and an optional adverb, succeeded by an optional adverb and a preposition”. It covers phrases such as “… was not activated by …”. This is a rather broad definition because there are only a handful of verbs and preposition that can be used as the VBD and IN. Similarly, active verb forms are defined as `ACT: {<MD>*<VB.?>*<RB.?>*<VB.?><RB.?>*}`, which reads “any tense of verb preceded by an optional modal, helping verb, an adverb and succeeded by an optional adverb”. It covers cases such as “… could only bind dATP”.

The training and testing data are split into 8:2 ratio randomly each time the code is run. Thus, it is difficult to state a fixed value for precision, recall and f-score. However, I could say that the precision is almost always 1.0, recall ranges from 0.4 to 0.6, while the f-score ranges from 0.5 to 0.7. Achieving 1.0 for precision was relatively easy because in almost all of the cases in which the set of triples is empty, the extractor did not extract any triples, which minimized false-positives. The formula for calculating precision revealed that 0 for false-positive was the reason that the precision was 1. Most of the “no triples” negative cases was because the action word was used as an adjective such as “Sldc-activated proteins” or as gerunds such as “Increasing A was …”. The grammar rules excluded such cases, effectively lowering the false-positives.

Possible Improvements

The most difficult issue to address was that the RegexpParser missed some of the rules defined. For example, in “A6 binds to CD44 resulting in the inhibition of migration, invasion, and metastasis of tumor cells, and the modulation of CD44-mediated cell signaling.”, “the inhibition of migration, invasion, and metastasis” is made into one noun phrase tree, and “tumor cells, and the modulation of CD44-mediated cell signaling” is made into another. The grouping of the two noun phrases is strange, but what is more mysterious is that those two are not made into a single noun phrase tree, although it is a valid form of the sixth rule of noun grammar. It was difficult to solve this issue because it does not always appear even in similar sentences. If I could improve the possible limitations of RegexpParser, the results could have been more accurate.

Another shortcoming was that the POS tagger is not always correct, so it confuses the RegexpParser. For example, in “HNRNPA2B1 directly binds a set of nuclear transcripts and elicits similar alternative splicing effects…”, “elicits” was tagged as a plural noun (NNS), which led the extractor to think that “a set of nuclear transcripts and elicits” is a noun phrase. I believe this is more of an inherent limitations of POS taggers and more difficult to solve than the previous issue.

Lastly, there were some exceptional cases in which the grammar rule was applied correctly but should not have been. For example, in “In addition, DST inhibited hepatic lipid accumulation…”, “addition, DST” was grouped as a noun phrase. This is an application of the fourth noun grammar, but obviously should have been avoided. I could not think of a programmable method to address this issue, but if I could, the results would have improved significantly.