In the Lincoln file, there are 653 sentences. Among them, we completely failed to parse four of them. In the rest sentences, there are 349 of them that we cannot find a subject, a predicate, or an object for the sentence. Since there are some sentences that do not contain one of them, we decided to pick some examples and check them manually. We randomly picked 10 sentences that contain a not existing part and 10 sentences that parsed everything. We found out that 7 of the first kind is correct and 10 of the second kind is correct. With the same ratio, we assume that we parsed 544 sentences correct in total (which is roughly 83.3% of total sentences).

Whether a sentence can be parsed by our parser depends on the root we found from the sentence. For most of the sentences, the root will be the main verb of it. So we handle all kinds of verbs and find the subject and object of it accordingly. However, if the verb is in passive form, the word can only be recognized as it is in past participle form. Therefore, we cannot check whether a verb is in passive form or not. The parsed result will have the subjective as objective and objective as subjective. Some sentences will have a noun as the root. It will be the subject of the sentences and we can find the subject and the predicate (usually the predicate will be copula, otherwise it will be the root) of it.

For three of the sentences we failed to parse, the root of it is “CD”. For example, for the sentence “As Douglas and the other candidates went through with their campaigns, Lincoln was the only one of them who gave no speeches.”, the parser claim that the root is “one”. We assume it is the tag for numbers. Since the number can occur at almost any place in the sentence, we did not find a good way to parse it. So the sentences that have a “CD” kind as they cannot be parsed.

There is also one sentence that has a root as “RB” and we failed to parse it. “Later that spring, Denton Offutt, a New Salem merchant, hired Lincoln and some friends to take goods by flatboat from New Salem to New Orleans via the Sangamon, Illinois, and Mississippi rivers.” In this sentence, the root is “Later”. Because this is the only case where the root has tag “RB” when the verb exist in the sentence, we decide not to handle this edge case.

Our parser also does not work well on sentences that contain clauses. It can often figure out some elements of the sentence, but sometimes it messes up in the clause or just cannot find the elements. Our parser works well on most of the conjunction sentences. It will figure out the conjunctive word, like and, or, but, and parse the remaining parts again. The starting code gives us a case when a “DT” will be found as the root. However, we cannot come up with an example and does not saw the situation when parsing Lincoln document.

To parse a given sentence, our parser will first use the Stanford NLP library to figure out the tree structure of the sentence and return the root of it. The two main cases are nouns and verbs. For nouns that are many different kinds for it, like “NNS”, “NNP”. These nouns have different names for their types according to some subtle differences, like whether it is in plural form, but they all act as the same role in a sentence. So we can just check if the name of the type begins with “NN”. Verbs are similar in this way. They vary according to their tense, but they all share the tag that begins with “VB”. So we just check if they begin with “VB”. The other situation we found is that if the verb is “am/is/are” and the sentence structure is simple. For example, when we parse “That man is handsome”, the root of the sentence structure will be “handsome” and the name of the type is “JJ”. We guess that this tag is for all the adjectives. In this case, we assume the root to be the object and we handle them the same as nouns.

After we figured out the root (and its type) of the sentence, we handle each case differently: When the root is a noun, we will find the word with tag “cop” (copula) under the root because it is guaranteed to be the predicate of the sentence. We will also find the word contains tag “nsubj” under the root because it will be the object. We then treat our root as the objective. Sometimes, the sentence might not have a subject or predicate. So if it is null, we will simply say the subject or predicate does not exist.

When the root is a verb, we will perform a more comprehensive analysis (since the sentence structure would be more complex). When the sentence is simple, we only need to find the words with tag “nsubj” and word with tag “obj” and they will be our subject and objective.

For the compound sentences, there must be a conjunctive word in the sentence. Otherwise, it will be a run-on sentence. So we just try to find whether there are verbs with tag “conj” under the root. If there is, we would parse the compound part(s) again to get the information of the compounded part. If we cannot find a subjective in the compound part of the sentence, we will assume that the subjective we found from the main part of the sentence is also subjective of this compound part.

After we figured out the subjective(s), predicate(s) and objective(s), we use the structure Triple provided by Stanford NLP to store the values we obtained. Then, we create a hashmap, mapping from the sentence to the list of Triple(s) associated with it. The key of the map is the sentence itself. The value is a list of Triples.

Our search procedure is quite straightforward. We will first parse the input question. The parse result of a sentence for the question is the same as the sentence of a statement. The result of the parsed question will also be stored in a Triple. If the question cannot be parsed successfully, it will answer “No” to the question. During the process of the document, we have a hashmap that contains all the Triples in it. We will just go through the Triples to see if there is a triple that has all of its three elements matched with the question’s Triple. If yes, we will answer “Yes” to the question. Otherwise, we will answer “No”. Since the nouns and verbs might be stored in slightly different ways because of the sentence structure, we used contains from both directions to check if they are the same word. All the characters have already been changed to lowercase during parsing. This will include most of the cases. However, if the verbs are in different tense and the verb has its special way of changing the tense, it will fail to detect the word.

We believe that our parser works well for most sentences. It does not have complex loops when parsing the sentences since we only focus on the root and its conjunctions. When dealing with the questions, we only parse the question once and then match the subjectives, predicates, and objectives. It is impossible for our parser to get stuck in some place. Therefore, the time performance of our parser should be acceptable. One main possible improvement for it will be to handle the sentences with clauses. Another improvement is to handle the questions answering better.