

Natural Language Processing

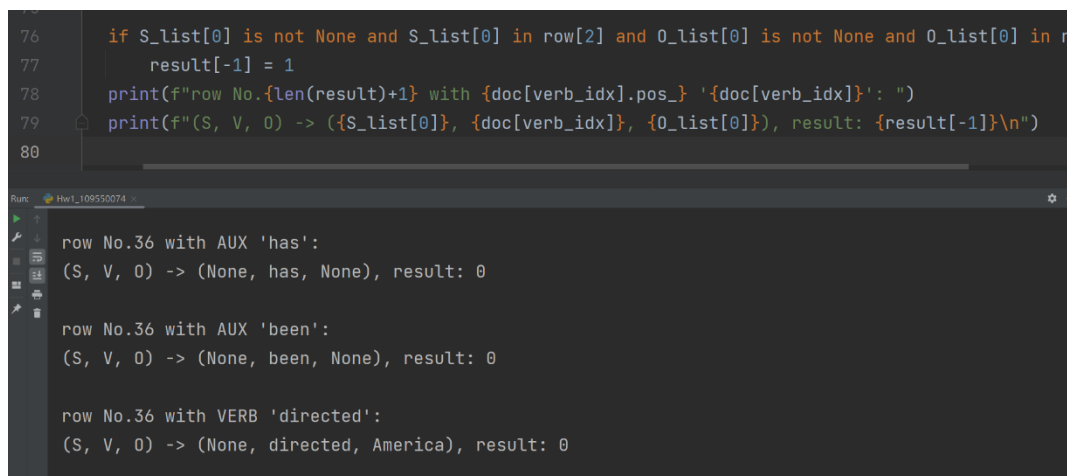
HW1 Report

1. Describing your methods in detail. (50%)

First, we use `csv.reader()` to read the data row by row, and use spaCy's "en_core_web_sm" as the model in this assignment since there are no too much improvement of accuracy in "en_core_web_lg" while the size of model is about 46 times larger according to spaCy's website.

Then, for each row in "dataset.csv", we use model to process the whole sentence, and give the return value the name 'doc', which may be referred to in the following paragraph. Now, we can start to check whether the given (S, V, O) relation can be extracted from the sentence or not. (Note that we don't need to extract all the possible relation in the sentence. Since the answer for each row is True or False, we only need to validate the relation in the given (S, V, O) tuple.

We start from the 'V' column in the dataset since that the VERB are usually the head of subject and object in 'doc'. However, not every sentence has a verb, for example: "It is a dog" only have 'is' in the sentence, which is auxiliary. Thus, when we want to find the verb in 'V' by POS tag, we also need to take the tag 'AUX' into consideration. You may question that this step could increase false positive rate, but the result shows that auxiliary like 'could' or 'will' almost can't find it's 'S' and 'O'.



```
76     if S_list[0] is not None and S_list[0] in row[2] and O_list[0] is not None and O_list[0] in row[2]:
77         result[-1] = 1
78     print(f"row No.{len(result)+1} with {doc[verb_idx].pos_} '{doc[verb_idx]}': ")
79     print(f"(S, V, O) -> ({S_list[0]}, {doc[verb_idx]}, {O_list[0]}), result: {result[-1]}\n")
80
```

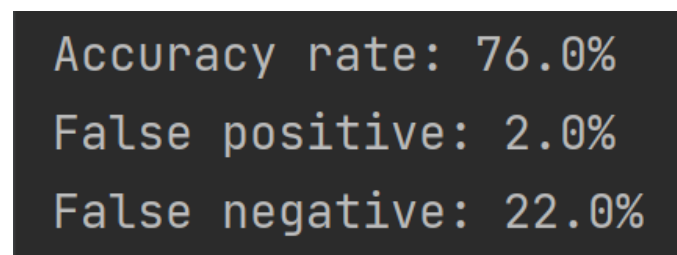
```
row No.36 with AUX 'has':
(S, V, O) -> (None, has, None), result: 0

row No.36 with AUX 'been':
(S, V, O) -> (None, been, None), result: 0

row No.36 with VERB 'directed':
(S, V, O) -> (None, directed, America), result: 0
```

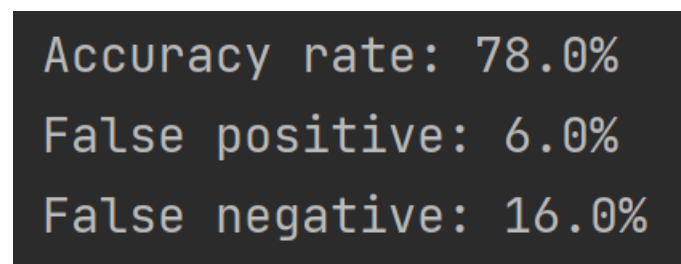
Figure 1. Part of the result of extraction and corresponding code

The remaining task is to find the subject and object for each verb in 'V'. Generally speaking, the three substring 'S', 'V' and 'O' will not overlap each other base on my observe in "example_with_answer.csv", so we can find the subject from "doc[(index of the first word in 'V') - 1]" back to doc[0]. Similarly, we can find the object from "doc[(index of the last word in 'V') + 1]" to doc[-1]. Although 'V' can be the rearranged substring of sentence and led to a lower accuracy rate than the normal search while using the "example_with_answer.csv" dataset, it give better result in Kaggle public test.



```
Accuracy rate: 76.0%  
False positive: 2.0%  
False negative: 22.0%
```

Figure 2. Accuracy rate if we use the search method mentioned above



```
Accuracy rate: 78.0%  
False positive: 6.0%  
False negative: 16.0%
```

Figure 3. Accuracy rate if we use the normal search method

Now the problem is that 'V' could contain some words even not in the sentence. We know that because some errors occurred when dealing with "dataset.csv", and that 'V' seems totally irrelevant to the sentence, so we decide to check if every word in 'V' is in the sentence beforehand. If not, we will set the prediction result to 0 immediately and move to the next sentence.

Finally, we have to find subject and object. For the subject, if the extracted relation is in the clause, we need to find the real subject instead of relative pronoun like 'who', 'which', 'that', 'where'...etc. For object, if the entity is followed by a clause, we also need to remove it so that it can be contained in the 'O' column. Moreover, verb may not directly be the head of the object, just like the example "I went to

Japan”, the head of ‘Japan’ is ‘to’ while the head of ‘to’ is ‘went’. Therefore, we define “the layer of head”. For example, ‘went’ is the layer2 head of ‘Japan’ in “I went to Japan”. We allow up to layer3 head, like ‘see’ and ‘pencil’ in “I see dozens of pencils”, and this is the core method to achieve simple baseline in our implementation, I think.

When all the above process finished, we can start prediction. For each verb, if its subject is in ‘S’ and its object is in ‘O’, and both subject and object not ‘None’, then we will assign 1 to the result. Otherwise, keep the result value 0, which is the default value. Since each ‘V’ may contain multiple verbs, the prediction is 1 if any of verb in the same row can give the result 1.

At the end, I want to mention some tricky method to reach my highest score. Since there are some objects in “A, B, and C” form while ‘O’ column only contains B, we can reverse the way to think: If object in ‘O’ or ‘O’ in object, pass the test. Use this strategy in subject and ‘S’ as well. Eventually, the method to remove the clause in object can also give up due to object containing ‘O’ can now pass the test as well.

2. Is there any difference between your expectations and the results? Why? (20%)

Yes. Although the result almost become my expectation in training stage, it indeed makes me confuse when I submit my prediction on Kaggle first time. Before my first submission, my accuracy rate on “example_with_answer.csv” can reach 82%, however, public score only reaches 0.67828 in my first try.

There are two possible reasons. First, I design a rule which overfitting “example_with_answer.csv”. If ‘V’ is not the substring of sentence, I immediately set the prediction to 0 and run the next loop. After removing that rule, I get the 78% accuracy rate while Kaggle’s public score reaches 0.68304. So, training dataset and testing dataset is quite different even in this task.

Second, we have serious bias in training dataset. There are only 50 sentences to train but over 1200 sentences to test. Furthermore, something different in “example_with_answer.csv” and “dataset.csv”. Occasionally, the unsplittable verb phrase inevitably led to false prediction in “example_with_answer.csv”, which appear

in “dataset.csv” very often. Thus, I design a search method mentioned in the first question to get the public score 0.71315.

3. What difficulties did you encounter in this assignment?

How did you solve it? (30%)

The big problem I encountered was the high false negative rate. To fix this problem, I look at each sentence and ground truth of “example_with_answer.csv” manually again and again. But simply look at sentences cannot find any solution, so I write a function to print the text, POS tag, dependency, head’s text, and subtree’s list of each word in the given sentence. When I found the parsing result of subject and object with given verb is different from my expectation, I can simply call the function and come out with strategy to reduce the false negative rate by observing the output, without messing up the main part of the program. Another method I’ve tried to reduce false positive rate is to use “nltk.tree”, which can draw a beautiful tree structure of dependency parsing, allowing me to see what condition is too strict so that the prediction tend to give the 0 result.

Nevertheless, the “MVP” is still the function to present accuracy rate, false positive rate, and false negative rate. If I hadn’t known these statistic beforehand, I can’t even start to reduce false positive rate.