Assignment 3

-yixi rao u6826541

Q2A

Return value: True
Dependencies: [(0, 1), (1, 2)]
Return value: True
Dependencies: [(2, 1), (3, 2), (0, 3)]
Return value: True
Dependencies: [(0, 1), (1, 2), (2, 3)]
Return value: True
Dependencies: [(0, 1), (1, 2), (2, 3), (3, 4), (4, 5)]
Return value: True
Dependencies: [(2, 1), (0, 2)]

Q2B

Action: ['right', 'right']
Features: [['[root]_stack', 'a_buffer'], ['a_stack', 'b_buffer']]

Success case 1
Sentence: [ROOT] A man dressed in a tux holds a violin .
Actions: ['shift', 'left', 'shift', 'right', 'right', 'shift', 'left', 'right', 'reduce', 'reduce', 'reduce', 'left', 'right', 'shift', 'left', 'right', 'reduce', 'right']
Features: [['[root]_stack', 'a_buffer'], ['a_stack', 'man_buffer'], ['[root]_stack', 'man_buffer']] ....

Success case 2
Sentence: [ROOT] A small cheese pizza is hot on the pan .
Actions: ['shift', 'shift', 'shift', 'left', 'left', 'left', 'shift', 'left', 'right', 'right', 'reduce', 'right', 'shift', 'left', 'right', 'reduce', 'reduce', 'right']
Features: [['[root]_stack', 'a_buffer'], ['a_stack', 'small_buffer'], ['small_stack', 'cheese_buffer']] ....

Success case 3
Sentence: [ROOT] A woman at a restaurant sits in front of a finished plate .
Actions: ['shift', 'left', 'shift', 'right', 'shift', 'left', 'right', 'reduce', 'reduce', 'left', 'right', 'right', 'right', 'right', 'shift', 'shift', 'left', 'left', 'right', 'reduce', 'reduce', 'reduce', 'reduce', 'right']
Features: [['[root]_stack', 'a_buffer'], ['a_stack', 'woman_buffer'], ['[root]_stack', 'woman_buffer']] ....

Fail case 1

['[ROOT]', 'A', 'doughnut', 'that', 'a', 'person', 'is', 'using', 'to', 'see', 'out', 'of', '.']
Actions: ['shift', 'left', 'right', 'shift', 'shift', 'left', 'shift', 'shift', 'left', 'left', 'shift', 'shift', 'left', 'right', 'right', 'right', 'reduce', 'reduce', 'reduce', 'shift']
Features: [['[root]_stack', 'a_buffer'], ['a_stack', 'doughnut_buffer'], ['[root]_stack', 'doughnut_buffer']] ....
Parser Dependencies:('[ROOT]', 'doughnut')('doughnut', 'A')('person', 'a')('using', 'person')('using', 'is')('using', 'see')('see', 'to')('see', 'out')('out', 'of')
Ground Truth Dependencies:('[ROOT]', 'doughnut')('doughnut', 'A')('doughnut', 'using')('doughnut', '.')('person', 'a')('using', 'person')('using', 'is')('using', 'see')('see', 'that')('see', 'to')('see', 'out')('out', 'of')

Fail case 2
['[ROOT]', 'The', 'waters', 'are', 'very', 'choppy', 'today', 'which', 'is', 'making', 'it', 'hard', 'to', 'ride', '.']
Actions: ['shift', 'left', 'shift', 'left', 'right', 'shift', 'left', 'right', 'reduce', 'right', 'shift', 'shift', 'left', 'left', 'reduce', 'reduce', 'shift', 'shift', 'left', 'right', 'shift', 'left', 'right', 'reduce', 'reduce', 'shift']
Features: [['[root]_stack', 'the_buffer'], ['the_stack', 'waters_buffer'], ['[root]_stack', 'waters_buffer']] ....
Parser Dependencies:('[ROOT]', 'are')('waters', 'The')('are', 'waters')('are', 'choppy')('are', 'today')('choppy', 'very')('making', 'which')('making', 'is')('making', 'hard')('hard', 'it')('hard', 'ride')('ride', 'to')
Ground Truth Dependencies:('[ROOT]', 'are')('waters', 'The')('waters', 'making')('are', 'waters')('are', 'choppy')('are', 'today')('are', '.')('choppy', 'very')('making', 'which')('making', 'is')('making', 'hard')('hard', 'it')('hard', 'ride')('ride', 'to')

Fail case 3
['[ROOT]', 'A', 'woman', 'holding', 'three', 'bags', 'stands', 'by', 'the', 'open', 'doors', 'of', 'a', 'subway', 'car', ',', 'through', 'which', 'other', 'people', 'can', 'be', 'seen', '.']
Actions: ['shift', 'left', 'shift', 'right', 'shift', 'left', 'right', 'reduce', 'reduce', 'left', 'right', 'right', 'shift', 'shift', 'left', 'left', 'right', 'right', 'shift', 'shift', 'left', 'left', 'right', 'reduce', 'reduce', 'reduce', 'reduce', 'right', 'shift', 'right', 'shift', 'left', 'shift', 'shift', 'shift', 'left', 'left', 'left', 'reduce', 'left', 'reduce', 'reduce', 'shift', 'shift']
Features: [['[root]_stack', 'a_buffer'], ['a_stack', 'woman_buffer'], ['[root]_stack', 'woman_buffer']] ....
Parser Dependencies:('[ROOT]', 'stands')('woman', 'A')('woman', 'holding')('holding', 'bags')('bags', 'three')('stands', 'woman')('stands', 'by')('stands', ',')('by', 'doors')('doors', 'the')('doors', 'open')('doors', 'of')('of', 'car')('car', 'a')('car', 'subway')('through', 'which')('people', 'other')('seen', 'through')('seen', 'people')('seen', 'can')('seen', 'be')
Ground Truth Dependencies:('[ROOT]', 'stands')('woman', 'A')('woman', 'holding')('woman', 'seen')('holding', 'bags')('bags', 'three')('stands', 'woman')('stands', 'by')('stands', ',')('stands', '.')('by', 'doors')('doors', 'the')('doors', 'open')('doors', 'of')('of', 'car')('car', 'a')('car', 'subway')('through', 'which')('people', 'other')('seen', 'through')('seen', 'people')('seen', 'can')('seen', 'be')

Total fails: 9, Total success: 19991

Reason: because the three failed cases' gold tree are non-projective, and according to the Yoav and Joakim's description of the Static Oracles for Transition-Based Parsing, which said "This algorithm is provably correct in the sense that, for every sentence x and projective dependency tree $G_{gold} = (V_x, A_{gold})$", so the **Algorithm 1** can only produce a sequence of actions that can produce a projective tree. On the other hand, $parse\_gold$ function is implemented based on the arc-eager transition Parsing, which is complete (but not sound) for the class of projective dependency trees.

Q3

1. What pre-processing did you do?

The first thing I need to do is to identify the positive samples and the negative samples. And we only care about the nationality relation in the dataset, so I extract the samples that have the nationality relation and give it label '1' to be positive sample, and for other samples which do not have the nationality relation but have the PERSON and GPE entities, I tag it '0' as the negative sample. Next, I trim the sentence to obtain a smaller segment that contains the PERSON and GPE entities, the words between the entities, and a few words before and after the entities. The purpose of the trimming operation is to remove the parts of the sentence that are not relevant to the relation extraction. The window size is a hyperparameter. Then I do some Normalization, which includes removing the stop words, the digits and punctuation. They add no information about the relationship between entities in the sentence. Stemming is done in this step, because I do not want the size of the word space be very large, and we should focus on the actual meaning of the word. The last thing I need to do is to extract the POS information of each word in the sentence, this will be an important feature.

2. How did you split the data into training and validation sets?

We have positive samples and negative samples, the ratio of positive samples and negative sample is 2:3, so the training data is very balanced and no need to define the class weight. I split the data into training, validation, testing dataset in ratio 7:2:1, and the data is split in a stratified fashion to let training, validation, testing dataset have the same proportion of the positive and negative samples. The training dataset is used to train the logistic regression model, the validation dataset is used to tune the hyperparameters, and the testing dataset is used to give a final performance of the model.

3. How did you choose hyperparameters?

The hyperparameters in this model are the window size of the trimming process, the n-gram range, the different combinations of the features, the Inverse of regularization

strength C. The method I use to choose the hyperparameters with best performance is the grid search, I first focus on only one hyperparameter and let other hyperparameters be fixed, after finding the best focused hyperparameter, we move to the next one, and so on.

4. What features did you choose and why did you choose these features?

The first feature I selected is the Bag of words or bigrams between PERSON entity and GPE entity. The BOW can tell us what words are used and what the position of the words. However, the model does not consider the order of different words in the sentence, it only considers the occurrence of each word separately. As a result, I use the n-gram approach, which looks at consecutive sequences of words of length n, and treats each sequence of words of length n as a different word. Besides, I use the Words or bigrams in particular positions, specifically, for the window size = 2, it will extract two words before the PERSON entity, and two words after the GPE entity, the reason why I choose this feature is that the surrounding text of the entity might reveal some information of the relation, such as, for nationality relation, the word "from" or phrase "come from" often appear near the entities.

I also choose Syntactic-based feature such as the POS tags, the POS tag is useful because part of speech can tell us the grammatical structure based on both its definition and its context. And the context of the entity can tell us the relation of the entities.

5. What post-processing did you do?

We predict the relation between any pair of PERSON and GPE in the same sentence. For example, predict the relation of the pair of (Miller, United States) or (Miller, Canada). It is possible that the model classifies one person into multiple nationalities, such as Miller comes from US and also comes from Canada, which is probably wrong because the majority of people only have one nationality. Therefore, we need to post-processing the data to ensure that the same person has only one nationality. The approach I used is comparing the probability the tuple $(P, G_1), (P, G_2)$ where P is the same person and $G_1 \neq G_2$ by using the raw output of the logistic regression (e.g.$\Pr\big((P, G_1)\big) > \Pr\big((P, G_2)\big) \rightarrow return\ (P, G_1)$). So, for each person, I choose the highest probability of the GPE as the final nationality of that person. Another advantage of this process is that it can reduce the repeated pair of PERSON and GPE.