ECS763P - Natural Language Processing: Assignment -1

TANISHQ VERMA
210642458

Queen Mary University of London — November 22, 2021

Abstract

In this assignment, I optimized the Condition Random Field CRF sequence tagger for a dataset movie trivial question which consist of target classes/label for each word in BIO (Beginning, Inside, Outside) tagging format. It also consists of pre-processing, training, and developing tagger on training data.

1 Question 1

In this question, the dataset is split into two parts into training data and testing data with an 80-20 split ratio. This is done using import SKLEARN library with function train_test_split and training and testing has been done on the spitted dataset accordingly.
Number of training samples = 6252
Number of testing sample = 1564

2 Question 2

The classification report was stored in form of a dictionary to make sorting easy. The dictionary was then sorted by precision giving the five classes with the least precision. I calculated the false-positive corresponding to these five classes. False-positive was calculated by checking the ground truth and the predicted tag that are corresponding to these five classes. There is a precise illustration with the word, predicted tag, ground truth tag, index of word in the sentence, and sentence for easy understanding.

3 Question 3

In the above question, the classification report was already stored. The dictionary was then sorted by recall giving the five classes with the least recall. I calculated the false-negative corresponding to these five classes. The false-negative was calculated by checking the ground truth and the predicted tag that are not corresponding to these five classes. Like the earlier question, there is a precise illustration for easy understanding.

4 Question 4

A pos-tagger was introduced in the preprocessing so that the word generated be as word-pos-tagger. I also changed the getfeature function by adding token.split(), with an argument in the function as pos tag. Pre-processing along with training of train and test data was done, the result was then created as a classification report. While comparing the classification report with the above preprocessed without pos-tag, there was a slight difference. The precision micro-average was increased by .01 however, the recall micro-average was decreased by the same amount. There was no change in the macro-average f1-score.

5 Question 5

In this question, there are already certain features present like capitalization, numbers, punctuations, suffixes, and even pos-tag which were added from the previous question. All these features are important in training a dataset and have been helpful.

However, to increase the macro-average f score we need to add more features. I found certain features useful in training the dataset and help me improve the overall macro-average f score. These features include a hyphen, is first, is last, previous (word & tag), and next (word & tag). To illustrate the feature, let me take an example and explain. “Hitler started WW-2”. In this example, WW-2 will be considered a single word if I had not used hyphen. It will therefore not capture the complete meaning of the sentence. Similarly, it goes for the rest of the features as well. Using the is first and is last will help us determine the object and subject of the sentence. The same goes for the previous (word & tag) and next (word & tag). For example, consider the sentence “Show me and give me the communication and performance requirements”. If this sentence is changed to “show me the communication and give me the performance requirements”, then the whole meaning of the sentence will be changed. Hence, adding all features will help us in better training the dataset.

For the preprocessing, the same function defined in the pos-tag function has been used and there are no changes to it. As stated in the question, the whole dataset is added to training and testing data. In the training tagger, the ct model was changed by adding training_opt=”feature.minfreq”:2,”c1”:0.1,”c2”:0.5 which will further add minimum document frequency of length 2 with L-1 regularization and L-2 regularization as 0.1 and 0.5 respectively to avoid overfitting. Finally, training and testing were done and a classification report was printed.

Below table compare the performamce of model after adding all those features.

model	macro avg precision	macro avg recall	macro avg f1-score
without pos-tag	0.64	0.53	0.56
with pos-tag	0.65	0.52	0.55
with added features	0.86	0.78	0.81

By adding all the features, I managed to train and increase the macro-average when compared to pos-tag or without pos-tag.