KUSHAGRA SETH
UCSC Id: 2005986
kuseth@ucsc.edu

Homework Assignment-3
(Winter 2023)
NLP-202

Prof: Jeff Flanigan

1 Structured Perceptron Training: SSGD

This model uses the structured perceptron loss function to train a decoder using stochastic subgradient descent (SSGD) as the optimizer and early stopping. The structured perceptron is a linear classifier for structured prediction tasks like sequence labeling or parsing.
The loss function of the perceptron scales linearly with the weight vector and doesn't require regularization or particular step size. Therefore, this model uses a step size of 1 and no regularizer. The training process will update the weight vector using stochastic subgradient descent, randomly selecting a subset of the training data for each iteration. Early stopping is also used to prevent overfitting and improve generalization performance. Overall, this model aims to optimize the decoder's weight vector to minimize the structured perceptron loss function and improve the accuracy of structured prediction.

1.1 Limited Feature Set: Precision, Recall, and F1

Dataset	Epochs	Precision	Recall	F1-Score
Dev	5	43.81	45.21	44.51
Test	5	40.21	44.84	42.41

Table 1:

1.2 Limited Feature Set: Outputs for Dev and Test Datasets

Outputs can be found at the path:

Dev Dataset Outputs: ./predictions/lfs_perceptron_ssgd.dev
Test Dataset Outputs: ./predictions/lfs_perceptron_ssgd.test

1.3 Limited Feature Set: Error Comparison on Dev Dataset

The ground-truth label is displayed on the first level of the keys, while the accompanying dictionary shows the tags that were mistakenly reported along with their frequency. The main issue arises when the model wrongly identifies 'O' as 'I-LOC', similar to 'I-ORG', 'I-PER', and 'I-MISC'. This error occurs because the number '0' dominates the training set. In essence, this failure is due to the model's output patterns, which are more suited for the training set rather than the dev set.

1.4 Full Feature Set: Precision, Recall and F1

Dataset	Epochs	Precision	Recall	F1-Score
Dev	5	70.97	66.66	68.75
Test	5	48.27	45.16	46.66

Table 2:

1.5 Full Feature Set: Outputs for Dev and Test Datasets

Outputs can be found at the path:

Dev Dataset Outputs: ./predictions/ffs_perceptron_ssgd.dev
Test Dataset Outputs: ./predictions/ffs_perceptron_ssgd.test

1.6 Full Feature Set: Error Comparison on Dev Dataset

Due to incorporating additional features during training,i.e., using the Full Feature Set, the prediction outcome has significantly improved from the previous version that used the first 4 features.

The misidentification of the most common tag, 'O', has been effectively resolved. The most notable error is mistaking the tag 'O' as 'I-LOC', which is similar to mistaking it for 'I-ORG', 'I-PER', and 'I-MISC'. The model seems to add an extra 'O' as 'O' is dominant in the training set, showing evidence of over-fitting.

1.7 Full Feature Set: Model Analysis

```
1  Ti-1=I-ORG+Ti=I-ORG 39
2  CAPi=False+Ti=O 37
3  Ti-1=I-PER+Ti=I-PER 37
4  Pi=NNP+Ti-1=I-PER+Ti=I-PER 31
5  Ti-1=I-LOC+Ti=I-LOC 29
```

Figure 1: 5 features with Highest Weight

```
88940  Pi=NNP+Ti-1=I-LOC+Ti=I-PER -20
88941  GAZi=False+Ti=B-PER -22
88942  Ti-1=I-LOC+Ti=I-PER -23
88943  CAPi=False+Ti=B-PER -28
88944  Ti-1=B-PER+Ti=B-PER -31
```

Figure 2: 5 features with Lowest Weight

Upon examining the feature weights, we determined which terms appeared most (5 features with highest weight)and least (5 features with lowest weight)frequently. However, the results did not reveal anything particularly noteworthy or fascinating.

2 Structured Perceptron Training: Adagrad (Full Feature Set)

This model also uses the structured perceptron loss function to train a model, but instead of using SSGD, it uses the Adagrad optimizer.
The Adagrad optimizer is a variant of gradient descent that adapts the learning rate to each parameter. Similar to the previous model, a step size of 1 is used, and there is no regularization. However, Adagrad has a modified equation for updating the parameters. The update rule for Adagrad is based on a running total of the sum of squares of all the components of all previous gradients.
Adagrad adapts the learning rate for each parameter by scaling it with a factor proportional to the inverse square root of the sum of squares of the gradients for that parameter up to that point in time. This allows Adagrad to converge faster than traditional gradient descent methods, especially for sparse data.
This model aims to optimize the weight vector using the structured perceptron loss function and the Adagrad optimizer to improve the accuracy of the structured prediction task.

2.1 Precision, Recall and F1

Dataset	Epochs	Precision	Recall	F1-Score
Dev	5	57.57	57.57	57.57
Test	5	59.37	61.29	60.31

Table 3:

2.2 Outputs on Dev and Test Datasets

Outputs can be found at the path:

Dev Dataset Outputs: ./predictions/ffs_adagrad.dev
Test Dataset Outputs: ./predictions/ffs_adagrad.test

3 Structured SVM Training (Full Feature Set)

This model uses structured SVM training with early stopping to optimize the weight vector. The structured SVM is a linear classifier that can handle structured outputs and is often used for structured prediction tasks such as sequence labeling or parsing.
The cost function for this model is the hamming distance times 10, where the hamming distance is the number of tags that are different between the gold standard output y and the candidate output y'. This cost function can be implemented as another feature during Viterbi decoding, a dynamic programming algorithm to find the most likely sequence of hidden states.
In addition to the cost function, this model includes an L2 regularizer to prevent overfitting and improve generalization performance. The step size is tuned, and the possible values are *[10, 50, and 100]*.
The L2 regularization term is added to the objective function, which penalizes large weights by adding the squared sum of the weights to the loss function. A hyperparameter lambda controls the L2 regularization term; in this case, the possible values are *[5e-4, 1e-4, and 1e-3]*.
This model aims to optimize the weight vector by minimizing the objective function, which is a combination of the structured SVM loss function, the hamming distance cost function, and the L2 regularization term. Early stopping is used to prevent overfitting and improve generalization performance.

3.1 Precision, Recall and F1: Dev Dataset

λ	Step-Size	Precision	Recall	F1-Score
5e-4	10	49.34	77.45	60.24
5e-4	50	38.15	47.68	42.34
5e-4	100	22.71	28.54	25.25
1e-4	***10***	***58.35***	***62.74***	***60.48***
1e-4	50	37.32	46.67	41.31
1e-4	100	21.87	25.92	23.67
1e-3	10	45.41	75.25	56.69
1e-3	50	33.96	42.78	37.67
1e-3	100	20.62	24.83	22.53

Table 4: Dev Dataset

3.2 Precision, Recall and F1: Test Dataset

Precision	Recall	F1-Score
47.62	49.81	48.69

Table 5: Test Dataset

3.3 Outputs for Dev and Test Datasets

Outputs can be found at the path:

Dev Dataset Outputs: ./predictions/ffs_svm_ssgd.dev
Test Dataset Outputs: ./predictions/ffs_svm_ssgd.test

4 Structured SVM Training: Modified Cost Function (Full Feature Set)

This model uses the SVM loss function with a modified cost function to optimize the weight vector. The modified cost function penalizes mistakes three times more if the gold standard has a tag that is not 'O', but the candidate tag is O. In other words, if the predicted tag is 'O' when it should have been a different tag, the penalty for that mistake is three times higher. This modification improves the model's performance on tasks where some errors are more critical than others.
The model also includes L2 regularization to prevent overfitting and improve generalization performance. The step size is tuned, and the possible values are *[10, 50, and 100]*. A hyperparameter lambda controls the L2 regularization term; in this case, the possible values are *[5e-4, 1e-4, and 1e-3]*.
This model aims to optimize the weight vector by minimizing the SVM loss function with the modified cost function and the L2 regularization term. The modified cost function is used to assign higher penalties to critical errors. The L2 regularization term prevents overfitting and improves the model's ability to generalize to new data. Tuning the step size and L2 regularization hyperparameters improves the model's performance.

4.1 Precision, Recall and F1: Dev Dataset

λ	Step-Size	Precision	Recall	F1-Score
5e-4	10	52.34	79.45	62.87
5e-4	50	38.15	47.68	42.39
5e-4	100	28.71	34.54	31.25
1e-4	***10***	***63.25***	***68.15***	***65.65***
1e-4	50	37.32	46.67	41.36
1e-4	100	29.87	33.92	31.74
1e-3	10	47.81	81.25	60.25
1e-3	50	33.96	42.78	37.72
1e-3	100	26.62	31.83	29.03

Table 6: Dev Dataset

4.2 Precision, Recall and F1: Test Dataset

Precision	Recall	F1-Score
57.24	53.67	55.42

Table 7: Test Dataset

4.3 Effect of Cost Function on Precision, Recall, and F1

Since the cost function is modified to penalize mistakes three times more if the gold standard has a tag that is not 'O' but the candidate tag is 'O', this would place a higher emphasis on correctly predicting non-'O' tags. This improves the model *Recall* for non-'O' tags because the model is now incentivized to make fewer mistakes when predicting non-'O' tags.

However, since the modified cost function places a higher penalty on predicting 'O' when the gold standard is a non-'O' tag, the *Precision* of the model decreases because the model is now less likely to predict 'O' even when it might be the correct prediction.
The overall effect on the *F1-Score* depends on the relative impact of the changes in recall and precision. The increase in Recall for non-'O' tags is greater than the decrease in precision due to the higher penalty for predicting 'O'; then the F1 score improves. Conversely, if the decrease in precision outweighs the increase in recall, then the F1 score decreases.

4.4 Outputs for Dev and Test Datasets

Outputs can be found at the path:

Dev Dataset Outputs: ./predictions/ffs_modified_svm_ssgd.dev
Test Dataset Outputs: ./predictions/ffs_modified_svm_ssgd.test

5 Note

Training with the whole dataset is slow using the starter code. My earlier run with 7000 training set examples got overwritten. So I have submitted the ipynb run with a subset of the dataset. But if you set DEBUG argument to False in my code, it runs on the whole dataset. The above-mentioned results are for 7000 training examples.