Report

Aryan Chandramania

First, here are some sample usages of the model:

```
aryan% python pos_tagger.py

Do you want to load a pretrained model? (y/n): n
We will train a new model for you. Would you like to save it? (y/n): n
Do you want to use the default hyperparameters? (y/n): y
Using default hyperparameters
                Embedding dimension:   64
                Hidden dimension:      128
                Number of layers:      1
                Number of epochs:      10
Training model...

Epoch    Train Loss      Val Loss
================================
1       0.276578        0.081264
2       0.079976        0.068803
3       0.069404        0.066340
4       0.064741        0.066011
5       0.062000        0.067244
6       0.059797        0.066873
7       0.057432        0.067364
8       0.056381        0.066893
9       0.055352        0.066922
10      0.054621        0.068036

Test Loss: 0.002046

Classification report:
Accuracy: 0.9829707053469866
Precision: 0.9909615566932125
Recall: 0.9829707053469866
F1 Score: 0.9855553254497351

Your model has been trained :D
Do you want to try it out? (y/n): y
Enter a sentence: when is the flight

WORD    TAG
when    ADV
is      AUX
the     DET
flight  NOUN
You can enter another sentence or type 'exit' to quit:
exit
```

Figure 1:

```
aryan% python pos_tagger.py

Do you want to load a pretrained model? (y/n): n
We will train a new model for you. Would you like to save it? (y/n): y
What name would you like to save your model with? pos_tagger.pt
Do you want to use the default hyperparameters? (y/n): n
Enter embedding dimension: 128
Enter hidden dimension: 256
Enter number of layers: 1
Enter number of epochs: 10
Training model...

Epoch    Train Loss      Val Loss
================================
1       0.186306        0.069162
2       0.068931        0.064000
3       0.064010        0.066825
4       0.060623        0.065635
5       0.058702        0.068557
6       0.057789        0.066375
7       0.055680        0.066055
8       0.055099        0.067272
9       0.053965        0.067861
10      0.053602        0.068299

Test Loss: 0.002009

Classification report:
Accuracy: 0.9822952218430053
Precision: 0.9873355730944262
Recall: 0.9822952218430053
F1 Score: 0.9838411464104722

Trained a new model and saved it to pos_tagger.pt
Do you want to try it out? (y/n) y
Enter a sentence: when is the flight from mumbai to delhi

WORD    TAG
when    ADV
is      AUX
the     DET
flight  NOUN
from    ADP
mumbai  PROPN
to      ADP
delhi   PROPN
You can enter another sentence or type 'exit' to quit:
exit
```

Figure 2:

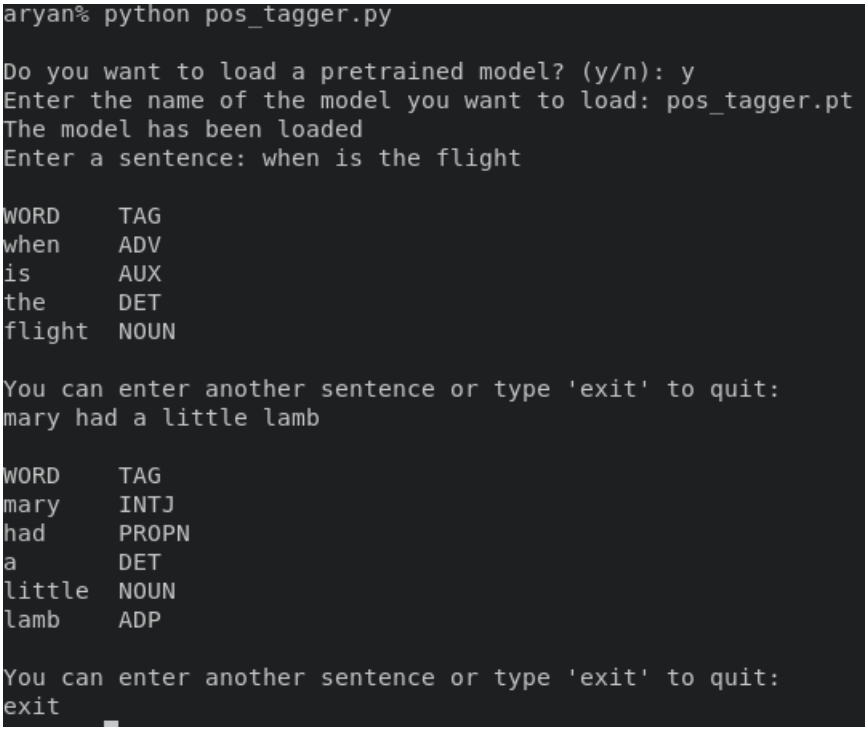
aryan% python pos_tagger.py
Do you want to load a pretrained model? (y/n): y
Enter the name of the model you want to load: pos_tagger.pt
The model has been loaded
Enter a sentence: when is the flight
WORD TAG
when ADV
is AUX
the DET
flight NOUN
You can enter another sentence or type 'exit' to quit:
mary had a little lamb
WORD TAG
mary INTJ
had PROPN
a DET
little NOUN
lamb ADP
You can enter another sentence or type 'exit' to quit:
exit

Figure 3:

I tried playing around with the hyperparameters a bit, and here are some of the results:

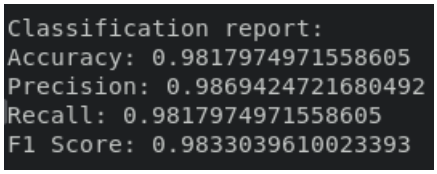

Figure 4: On running for 20 epochs with rest as default

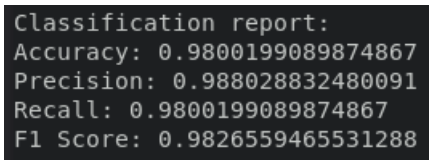

Figure 5: On running for 30 epochs with rest as default

As we can see, the number of epochs doesn't seem to really affect the accuracy of the model, but it gets just a bit worse.

As also seen in Figure 2, the accuracy remained roughly similar even when the embedding and hidden dimensions were doubled.

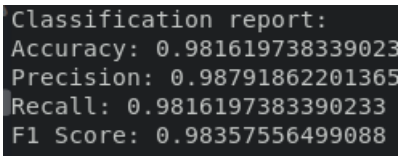

Figure 6: `embedding dim` = 128 with rest as default

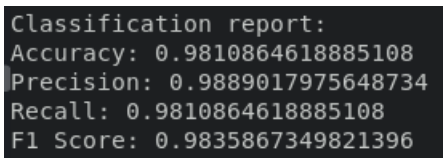

Figure 7: `hidden dim` = 256 with rest as default

As we can see, the accuracy of the model suffers only minor variations on changing the hyperparameters, and remains fairly consistent. In fact, since the batching involves shuffling, we are likely to see this extent of variation even for the same hyperparameter values.

Some other things to note:

- The model use batching, with a batch size of 32
- The model uses the Adam optimizer, with a learning rate of 0.01
- The learning rate wasn't tweaked.
- The loss function used is CrossEntropyLoss. Padding (which was necessary due to batching) was ignored while computing the loss.
- There is 1 hidden layer.
- The model is an LSTM model.
- The model took log softmax over the final layer.
- The script allows you to both train a new model and optionally save it, or load a previously saved model and test it.