NLP Report HW4

Aditi Pandurang Hande

Generic Steps irrespective of task 1 or 2

I have imported all the relevant libraries for the task.

Defined function NERDataset which takes file path and the word and label vocabs i have created and returns the tensors of sentence and labels.
This is later used to send data to the data loader.

For task1 my word vocab's zero index is for padding and first is to handle any unknown words. So these indices are fixed.

Word and label vocab essentially map word/label to a index. I have defined collate_fn function to pad the sentences as per batch, as model requires all inputs to be of same size.
In both tasks sentences are padded by 0 and labels by -1. This functions returns the padded sentences, length of sentence tensor and padded labels.

With defined batch size I have created dataloader for dev and train data.
For task one batch size is 32 and for two it is 8. These are found by permutation and one that gives best f1 scores.

I also have inverted_label_vocab which is just index mapped to label.

Task One

BLSTM class has the model architecture for bidirectional lstm.
All the parameters which are specified in pdf are passed to this.
Defined lstm,linear,dropout and elu layers. Embedding dimension in task one is 100. In embedding mentioned padding index as 0, as per the logic.

Architecture:

Embedding layer
Output of embedding layer to pack_padded_sequence to optimize computation by avoiding unnecessary calculations on padded values.
Output of that to the lstm layer to unpack the packed tensor and pads it with zeros to obtain a tensor of the original variable-length sequences.

Dropout layer
ELU /Activation layer
Then classifying activation's output using Linear (this is the final layer)

Hyper-params:

learning_rate = 1
num_epochs = 30
Found both of these by trial and error.

Used cross entropy loss with weights([1,0.5,1,1,1,1,1,1,1]) , assigned these by analyzing the per class f1 and recall and accuracy so that some classes are favored for prediction over others.
Passed ignore_index=-1 as labels were padded by -1 value.

Used SGD as mentioned in pdf.
Used lr_scheduler as just keeping one lr for all epochs was not giving decent results.
It is useful as it reduces the learning rate when the metric stops improving, so small steps when converging. Its params are mode='min',factor=0.5,patience=3,verbose=True.
Mode is mentioned as the metric is decreasing, if it doesn't change for 3 epochs, lr will be halved, verbose is mentioned to know when it's changing.

Trained the model for mentioned epochs and printed dev and train loss fo each, later saved the model as blstm1.pt

What are the precision, recall and F1 score on the dev data?

```
D:\NLP\HW4\hw4>perl conll03eval.txt < dev1.out
processed 50867 tokens with 5885 phrases; found: 5146 phrases; correct: 4337.
accuracy:  95.47%; precision:  84.28%; recall:  73.70%; FB1:  78.63
              LOC: precision:  93.04%; recall:  79.76%; FB1:  85.89  1567
             MISC: precision:  87.77%; recall:  75.55%; FB1:  81.20  785
              ORG: precision:  72.96%; recall:  65.99%; FB1:  69.30  1202
              PER: precision:  82.47%; recall:  72.30%; FB1:  77.05  1592
```

Task Two

Created glove_embedding which maps word to its embedding as per glove txt file given.
Created word_embedding such that every word with first letter capital, has its embedding appended by 1, if its present in glove if not, then unk's embedding appended by 1. This 1 symbolizes that the actual word is capitalized.

If the first letter is not capitalized then its embedding is appended by 0, if its present in glove if not, then unk's embedding appended by 0. This 0 symbolizes that the actual word is not capitalized.

BLSTM class has the model architecture for bidirectional lstm.
All the parameters which are specified in pdf are passed to this.
Defined lstm,linear,dropout dropout1 and 2 with 0.2 as p value and elu layers. Embedding dimension in task two is 101 (one more to accommodate the capitalized bit).
In the embedding mentioned padding index as 0, as per the logic and freeze as false so that the embedding learns as well.

.

Architecture:

Embedding layer
Output of embedding layer is given to dropout(p=0.2) (this gave better results as its regularized to prevent overfitting)
Output of dropout layer to pack_padded_sequence to optimize computation by avoiding unnecessary calculations on padded values.
Output of that to the lstm layer to unpack the packed tensor and pads it with zeros to obtain a tensor of the original variable-length sequences.
Dropout layer standard
Linear layer
Dropout layer of 0.2
ELU /Activation layer
Then classifying activation's output using Linear (this is the final layer)

Hyper-params:

learning_rate = 1
num_epochs = 50
Found both of these by trial and error.

Used cross entropy loss with weights ([1.2,0.5,1,1,1,1,1,1.2,1]), assigned these by analyzing the per class f1 and recall and accuracy so that some classes are favored for prediction over others (imbalanced dataset is the issue).
Passed ignore_index=-1 as labels were padded by -1 value.

Used SGD as mentioned in pdf.
Used lr_scheduler as just keeping one lr for all epochs was not giving decent results.

Tried multiple such as ReduceLROnPlateau, StepLR, OneCycleLR.
Best was StepLR, found empirically, it adjusts the learning rate of an optimizer by a fixed factor after a specified number of epochs. So in my case it applies a step function to the learning rate, reducing it by a factor of 0.5.

Trained the model for mentioned epochs and printed dev and train loss of each, later saved the model as blstm2.pt

What are the precision, recall and F1 score on the dev data?

```
D:\NLP\HW4\hw4>perl conll03eval.txt < dev2.out
processed 50638 tokens with 5862 phrases; found: 5514 phrases; correct: 4921.
accuracy:  97.13%; precision:  89.25%; recall:  83.95%; FB1:  86.52
              LOC: precision:  94.99%; recall:  89.77%; FB1:  92.31  1718
             MISC: precision:  77.53%; recall:  77.62%; FB1:  77.58  908
              ORG: precision:  86.20%; recall:  78.60%; FB1:  82.22  1210
              PER: precision:  91.90%; recall:  85.19%; FB1:  88.42  1678
```