NLP HW 4 – Gautami Langarkande

Task 1:

Data preprocessing –

For data preprocessing, I created imported the train data and created a dictionary called word2idx of unique words with indexes as values. I also added <PAD>:0 and <UNK>:1 to the dictionary. This is necessary to deal with unknown words.
I also created a dictionary for the nine NER tags and assigned an index to them called ner2idx. Then I created a nested list of sentences with the indexes of each word of the sentence by matching the words from the train data set and encoded them into tensors.
I did the same for the NER tags.

Model Training-

I built the model with the given architecture and assigned the following hyperparameters to the model.

num_epochs = 120
batch_size = 8

vocab_size = len(word2idx)
output_size = len(ner2idx) #9
embedding_dim = 100
hidden_dim = 256
num_layers = 1
dropout = 0.33
learning_rate = 0.5
weight = [0.7 , 1, 1, 1.5, 1, 1, 1.5, 1, 1.2]

I assigned weights to each tag to deal with data imbalance. I assigned weights to each tag to deal with data imbalance. I also added learning rate scheduling (OneCycleLR) to linearly decrease the learning rate as the training continues.
I trained the model for 120 epochs and got the following scores.
accuracy: 96.73%; precision: 88.85%; recall: 79.79%; F1 score: 84.08

Evaluation:

To evaluate the model, I imported the dev data set and did some preprocessing to convert the words into tensors and then predict the tags for the words. Then I used the script provided to calculate the F1 score

Task 2:

Data preprocessing –

For data preprocessing, I created imported the train data and created a dictionary called word2idx of unique words with indexes as values. I also added <PAD>:0 and <UNK>:1 to the dictionary. This is necessary to deal with unknown words.
I also created a dictionary for the nine NER tags and assigned an index to them called ner2idx. Then I created a nested list of sentences with the indexes of each word of the sentence by matching the words from the train data set and encoding them into tensors.
I did the same for the NER tags.

I imported the glove embeddings of 100 dim. And I created a function called is_capitalization() which checks if the word in train has the first letter capitalised or not.
After this, I check if the word in the data set has capitalization or not. If it does and if it exists in the glove embeddings then I assign the glove embeddings and I add and I concatenate 1 to the embeddings. I stored this value in a matrix.
Similarly, I check if the word does not have capitalization and if it exists in the glove embeddings then I assigned the glove embedding and concatenate 0 to the embeddings. If the word does not have a capitalization and if it does not exist in the glove embedding then I concatenate a 0 to a random array of size 100. If the word has capitalization and if it does not exist in the glove embedding then I concatenate a 1 to a random array of size 100.

Then like task 1, I convert this matrix into tensors. I also created tensor for the NER tags in the sentence just as I created in task 1.

Model Training-

I built the model with the given architecture. Additionally, as guided by TA Jun Yan on the piazza, I added an embedding dropout layer and a linear dropout layer to increase my accuracy in model training. The following hyperparameters to the model.

num_epochs = 50
batch_size = 32

output_size = len(ner2idx)
vocab_size = len(word2idx)
embedding_dim = 101
hidden_dim = 256
num_layers = 1
dropout = 0.33
learning_rate = 0.5
weight = [0.4 , 1, 1, 1.5, 1, 1.4, 1.5, 1, 1.2]

I assigned weights to each tag to deal with data imbalance. I also added learning rate scheduling (OneCycleLR) to linearly decrease the learning rate as the training continues.

I trained the model for 120 epochs and got the following scores.
accuracy: 97.48%; precision: 89.17%; recall: 86.72%; FB1: 87.93

Evaluation:

To evaluate the model, I imported the dev data set and did some preprocessing to convert the words into tensors and then predict the tags for the words. If a word with all capitalization is encountered, then it is considered a word with the first letter as capital. This is how I added an extra step to deal with the capitalization of all capitalized words. Then I used the provided script to calculate the F1 score.