Assignment 2

Report

The BiLSTM model consists of the following layers:

1. **`nn.Embedding`:** This layer takes in word indices and returns word embeddings of size (batch_size, sequence_length, wordEmbeddingDim). It uses pre-trained word embeddings as initialization.
2. **`nn.LSTM`:** This is a recurrent layer that takes in the word embeddings as input and returns output of size (batch_size, sequence_length, 2*lstmHiddenDimWord), where 2*lstmHiddenDimWord is the size of the concatenated hidden states of the forward and backward LSTM layers. The output is obtained by concatenating the hidden states of the forward and backward LSTM layers.
3. **`nn.Dropout`:** This is a regularization layer that randomly drops out a specified percentage of the input units during training to prevent overfitting. It takes in the output of the LSTM layer as input and returns output of the same size.
4. **`nn.Linear`:** This is a fully connected layer that takes in the output of the LSTM layer after dropout and returns output of size (batch_size, sequence_length, numUniqueTags), where numUniqueTags is the number of unique tags in the dataset.
5. **`F.log_softmax`:** This layer applies the log softmax function along the last dimension of the output of the linear layer to obtain the tag scores of size (batch_size, sequence_length, numUniqueTags).

The model takes in word indices as input and returns tag scores as output. During training, the model is optimized to minimize the cross-entropy loss between the predicted tag scores and the ground truth tags.

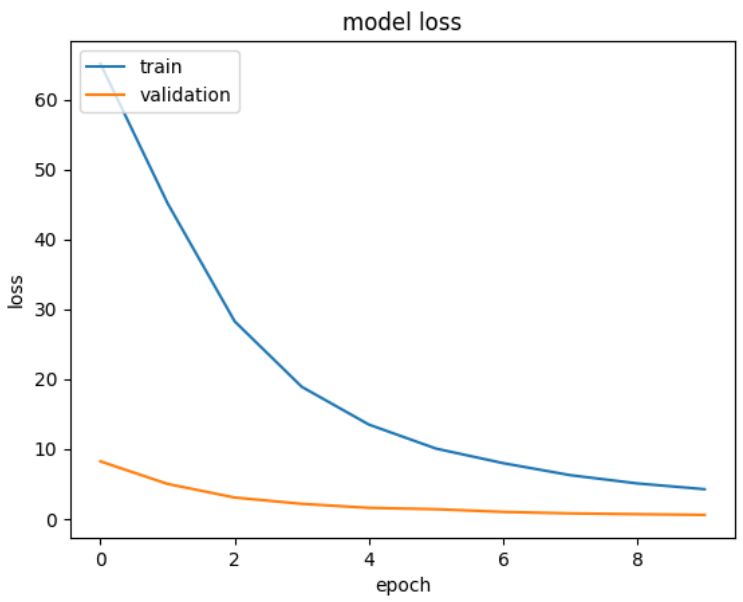
```
class BiLSTM(nn.Module):
    def __init__(self, numUniqueWords, numUniqueTags, wordEmbeddingDim, lstmHiddenDimWord,
dropoutRate,embedding):
        super(BiLSTM, self).__init__()
        self.wordEmbedding = nn.Embedding.from_pretrained(torch.from_numpy(embedding),
freeze=False,padding_idx=0)
        self.lstmWord = nn.LSTM(wordEmbeddingDim ,
                                lstmHiddenDimWord, bidirectional=True, batch_first=True)
        self.dropout = nn.Dropout(dropoutRate)
        self.linear = nn.Linear(2*lstmHiddenDimWord, numUniqueTags)

    def forward(self,word_indices):
        word_embeddings = self.wordEmbedding(word_indices)
        lstm_word_out = self.lstmWord(word_embeddings)[0]
        lstm_word_out = self.dropout(lstm_word_out)
        linear_out = self.linear(lstm_word_out)
        tag_scores = F.log_softmax(linear_out, dim=2)
```

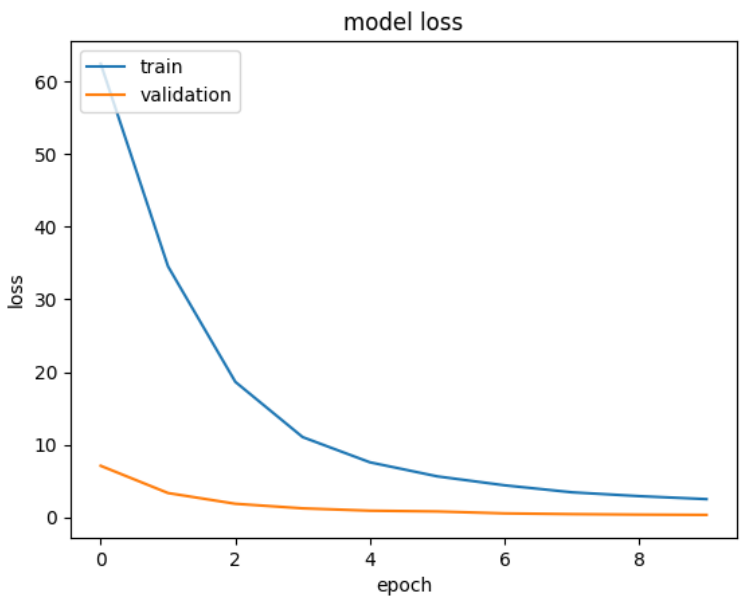
```
        return tag_scores
```

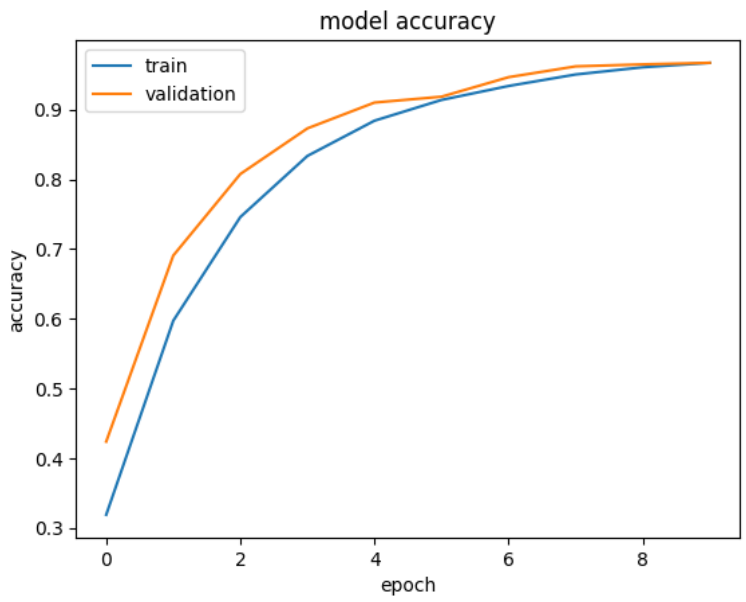
Hyperparameters:

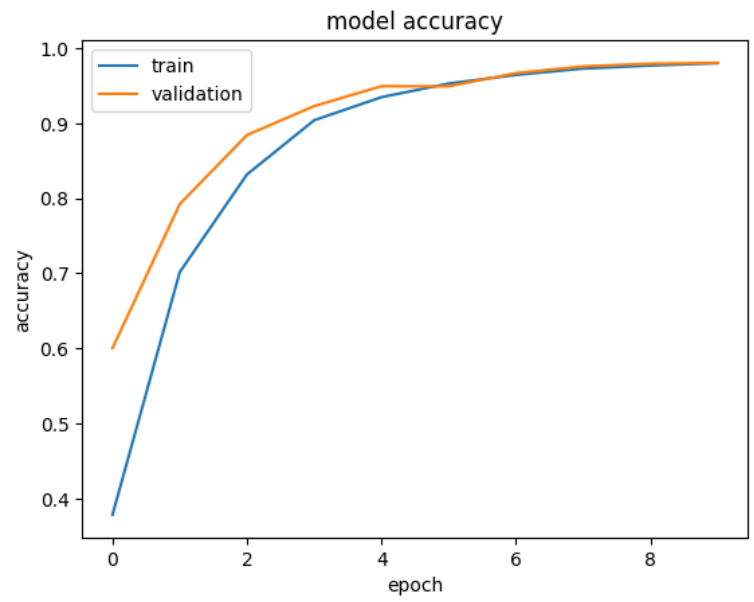
```
WORD_EMBEDDING_DIM = 100
LSTM_HIDDEN_DIM_WORD = 256
DROPOUT_RATE = 0.5
BATCH_SIZE = 128
NUM_EPOCHS = 10
LEARNING_RATE = 0.001
```

Word_EMBEDDING_DIM: Model's performance for different dimensions of pre-trained embedding used in the model. First Column is for dimension 50 and second for dimension 100. More higher dimensions can be tested (couldn't due to limited memory issue). It is clearly visible that with higher dimension of Word Embeddings , the model's performance is better.

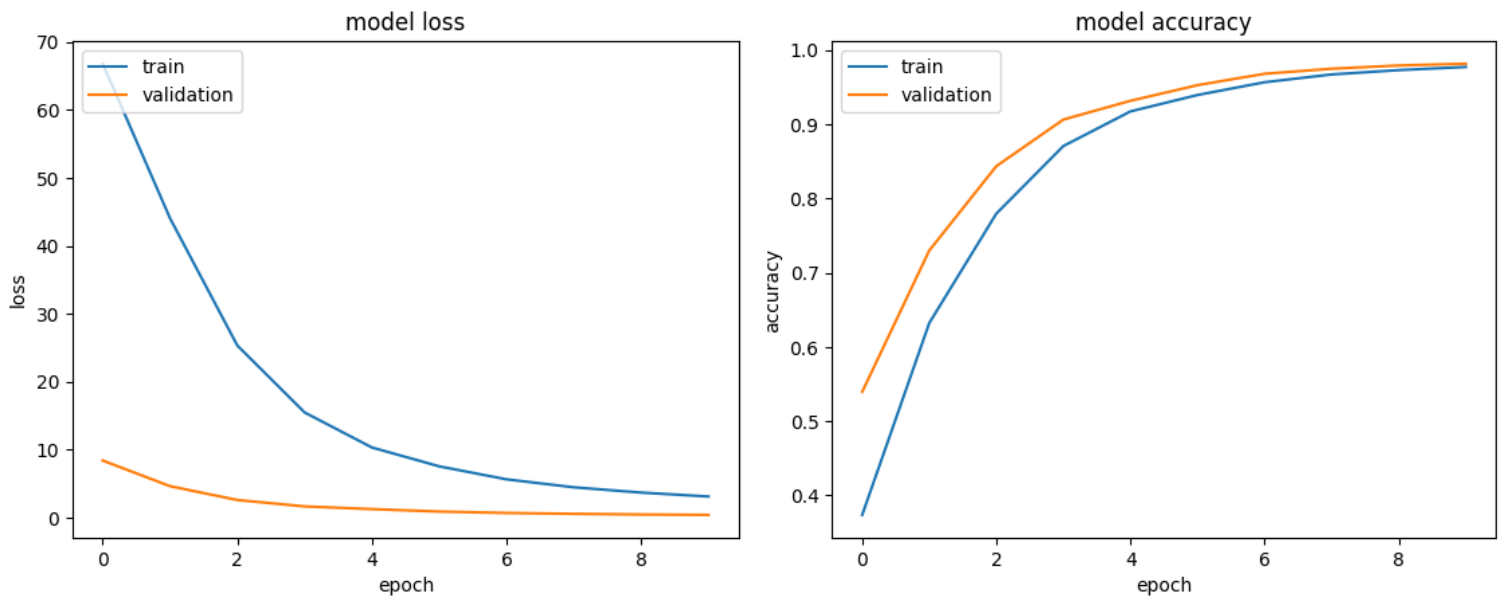


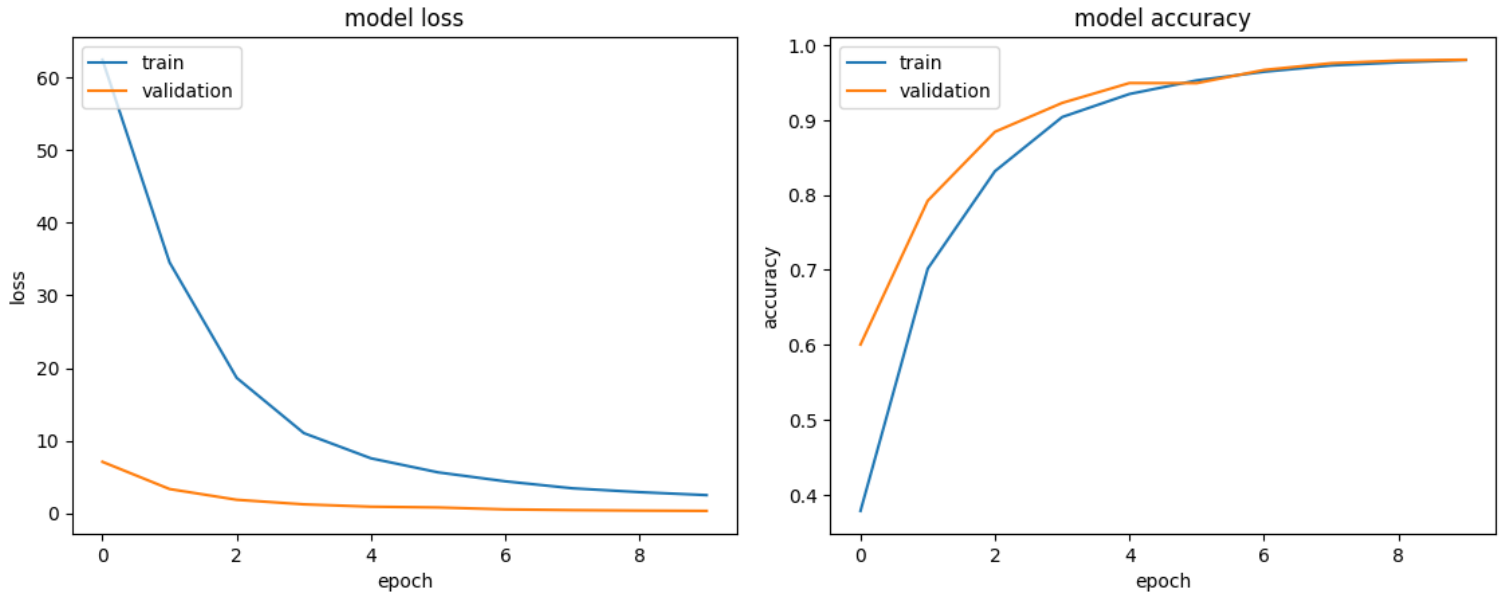


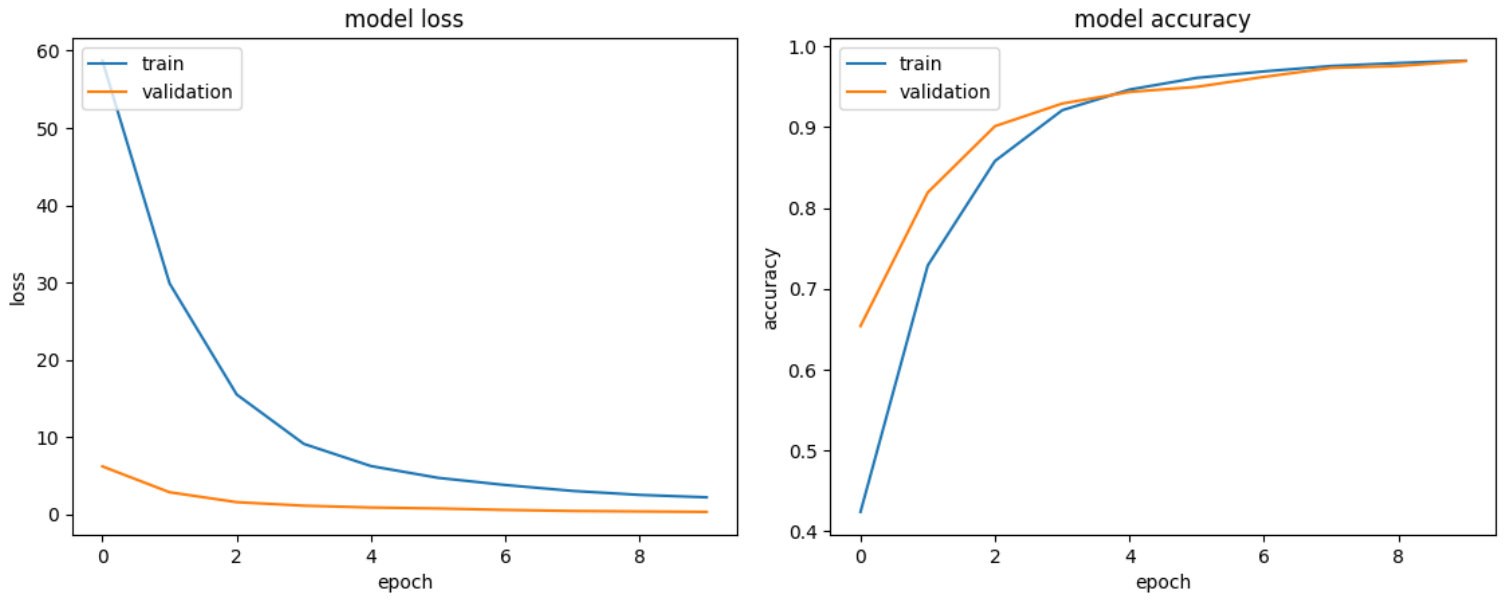




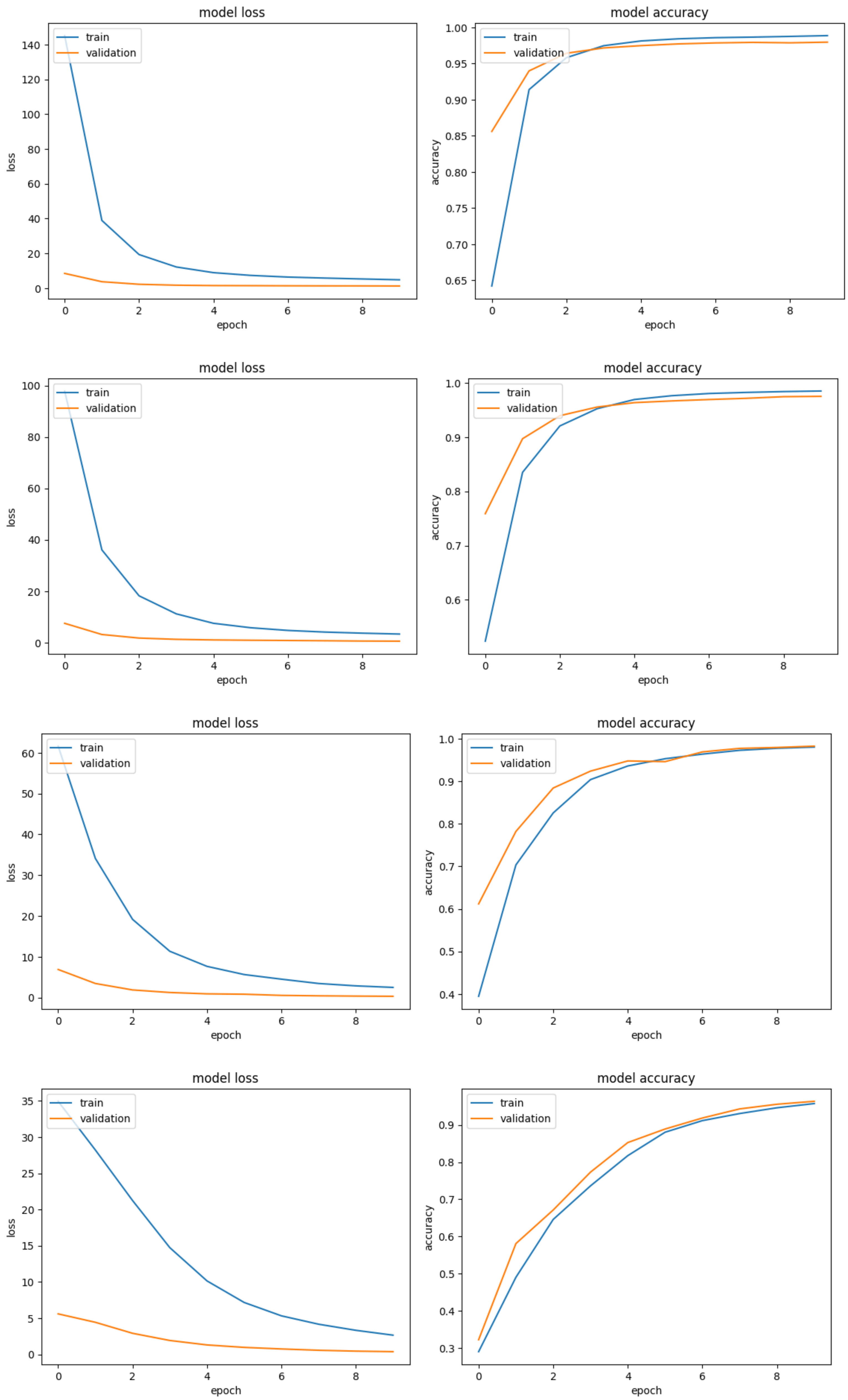
HIDDEN LAYER SIZE: Model's performance for three different hidden layer sizes (128,256,512). The first row is for 128, second row is for 256 and last row is for 512. It is clearly visible that taking 256 as hidden layer size has better performance.





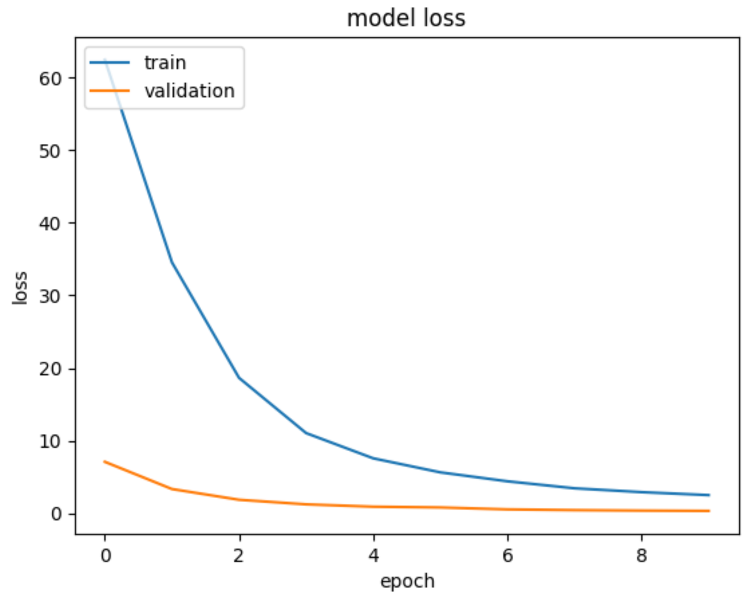


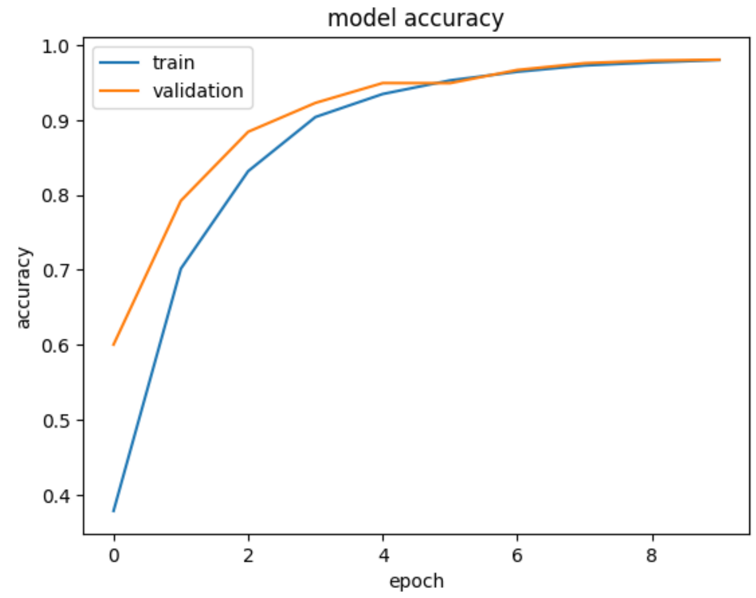
BATCH SIZE: Model's performance for 4 different batch sizes (32, 64,128, 256) with graphs in row-wise order. From the below graphs, 64 and 128 batch size perform better.

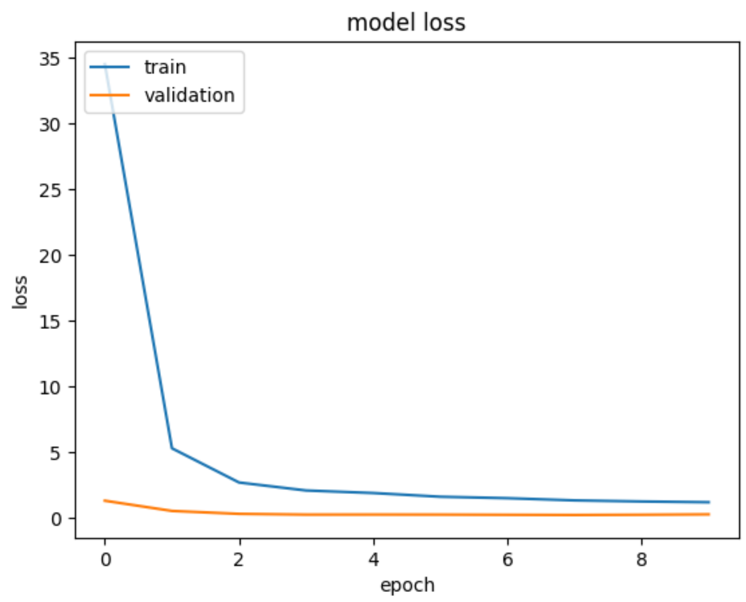


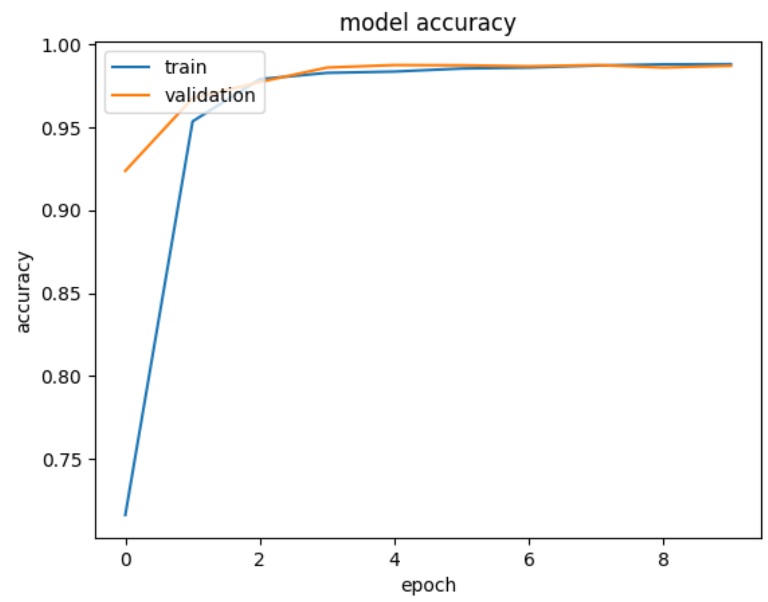
model loss
train
validation
loss
epoch
model accuracy
train
validation
accuracy
epoch
model loss
train
validation
loss
epoch
model accuracy
train
validation
accuracy
epoch
model loss
train
validation
loss
epoch
model accuracy
train
validation
accuracy
epoch
model loss
train
validation
loss
epoch
model accuracy
train
validation
accuracy
epoch

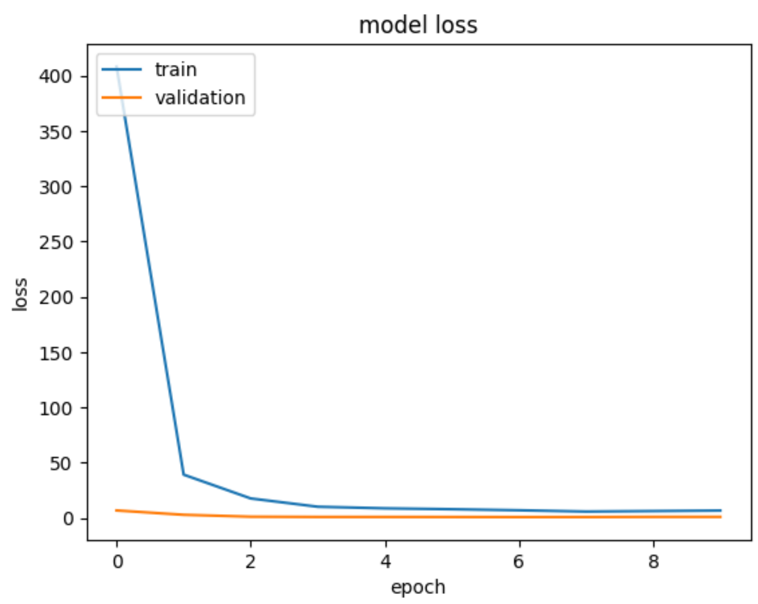
LEARNING RATE: For three different learning rates (0.001,0.01,0.1) . With learning rate = 0.01, the performance is very good.

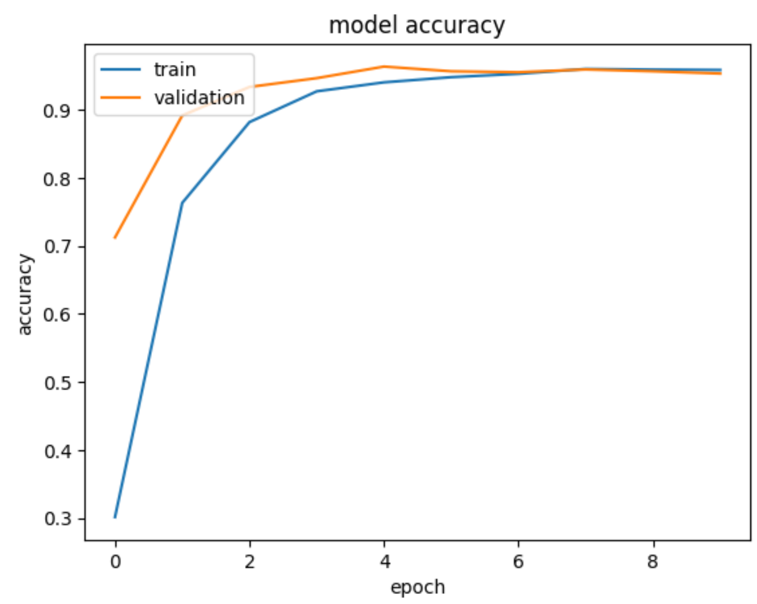












With **EMBEDDING_DIM** = 100, **HIDDEN_LAYER** = 256, **BATCH_SIZE** = 128 and **LEARNING_RATE** = 0.01, These are scores for **10** epochs.

Epoch	Training Loss	Training Accuracy	Validation Loss	Validation Accuracy
1	33.81175669282675	0.6728188264309938	1.3509207144379616	0.9202287778446718
2	5.567493926733732	0.9487822423183639	0.523823257535696	0.9661348585189645
3	2.685206539928913	0.9771863117870723	0.3104766942560673	0.9768211920529801
4	2.1301394943147898	0.9819340252800329	0.21398143097758293	0.9872065021071643
5	1.7779198866337538	0.9839276538896311	0.21075187996029854	0.9888621312462372
6	1.6777253430336714	0.9847703216524509	0.2282635997980833	0.9873570138470801
7	1.4959060717374086	0.9855924365430069	0.2290336899459362	0.9866044551475015
8	1.313508927822113	0.9872161134518549	0.21503868140280247	0.9872065021071643
9	1.268343790434301	0.9873599835577022	0.23106771893799305	0.9870559903672487
10	1.1850134981796145	0.9880587812146747	0.2408750895410776	0.9872065021071643

Classification Report (Precision, Recall, F1 score and Support) for Training and Testing respectively:

Training:

	precision	recall	f1-score	support
1	0.9997	0.9781	0.9888	3018
2	0.9988	0.9884	0.9936	1729
3	0.9855	0.9976	0.9915	3805
4	0.9965	0.9982	0.9974	8529
5	0.9963	0.9868	0.9916	10781
6	0.9988	0.9996	0.9992	11605
7	0.9620	0.9965	0.9789	4545
8	0.9992	1.0000	0.9996	1196
9	0.9779	0.9364	0.9567	1605
10	0.9987	0.9987	0.9987	750
11	0.9714	0.9098	0.9395	410
12	0.9073	0.9891	0.9464	366
13	0.9634	1.0000	0.9814	316
accuracy			0.9914	48655
macro avg	0.9812	0.9830	0.9818	48655
weighted avg	0.9916	0.9914	0.9914	48655

Testing:

	precision	recall	f1-score	support
1	0.9974	0.9898	0.9936	392
2	0.9961	0.9922	0.9941	256
3	0.9903	0.9961	0.9932	512
4	0.9957	0.9939	0.9948	1155
5	0.9965	0.9958	0.9962	1433
6	0.9942	0.9987	0.9965	1549
7	0.9935	0.9968	0.9952	618
8	0.9943	1.0000	0.9971	173

```
           9     0.9241    0.9495    0.9367       218
          10     1.0000    1.0000    1.0000       109
          11     0.8833    0.7260    0.7970        73
          12     0.9825    1.0000    0.9912        56
          13     1.0000    1.0000    1.0000        36

    accuracy                         0.9915      6580
   macro avg     0.9806    0.9722    0.9758      6580
weighted avg     0.9913    0.9915    0.9913      6580
```

Actual vs Predicted for Test set

```
Word                    Actual                  Predicted
what                    PRON                    PRON
are                     AUX                     AUX
the                     DET                     DET
coach                   NOUN                    PROPN
flights                 NOUN                    NOUN
between                 ADP                     ADP
dallas                  PROPN                   PROPN
and                     CCONJ                   CCONJ
baltimore               PROPN                   PROPN
leaving                 VERB                    VERB
august                  NOUN                    NOUN
tenth                   ADJ                     ADJ
and                     CCONJ                   CCONJ
returning               VERB                    VERB
august                  NOUN                    NOUN
today                   <UNK>                   <UNK>
i                       PRON                    PRON
want                    VERB                    VERB
a                       DET                     DET
flight                  NOUN                    NOUN
from                    ADP                     ADP
nashville               PROPN                   PROPN
to                      ADP                     ADP
seattle                 PROPN                   PROPN
that                    ADP                     ADP
arrives                 VERB                    VERB
no                      DET                     DET
```

Testing for Random Sentence

```
sentence = "We are coming from India"

OUTPUT:
            we          PRON
            are         AUX
            coming      VERB
            from        ADP
            india       <UNK>
```

"UNK" tag is given to words which are not present in the vocabulary with given minimum frequency.

Analysis

The task of part-of-speech (POS) tagging involves assigning a grammatical label to each word in a sentence, such as noun, verb, adjective, etc. In this case, a BiLSTM model was trained for POS tagging, which achieved high accuracy on both training and testing datasets.

The classification report shows precision, recall, and F1-score for each POS tag. The precision of a tag represents the proportion of correctly predicted instances of that tag among all instances predicted as that tag. Recall represents the proportion of correctly predicted instances of a tag among all instances that actually belong to that tag. F1-score is the harmonic mean of precision and recall, which gives an overall measure of the model's performance.

The results show high precision and recall for most tags, with an overall accuracy of 99.14% on the training data and 99.15% on the testing data. The model performs particularly well on tags 1(PRON), 4(NOUN), 5(ADP), 6(PROPN), and 7(VERB), which have precision and recall scores above 0.96 on both datasets. The model also achieves perfect precision and recall scores for tags 8(NUM), 10(CCONJ), 12(PART), and 13(INTJ) on the training data and tags 10(CCONJ) and 13(INTJ) on the testing data.

The model seems to struggle with classes 9 (ADJ), 11 (ADV), and 12(PART), as indicated by lower F1-scores for these classes on the testing set. This suggests that the model may have difficulty distinguishing between these classes, or that there may be some overlap between them in the training data.

The precision, recall, and F1-score for class 11 (ADV) are particularly low on the testing set, with an F1-score of only 0.797. This indicates that the model may have difficulty correctly identifying adjectives in some cases.

On the other hand, the precision, recall, and F1-score for class 10 (CCONJ) are perfect, with values of 1.0 for all three metrics on the testing set. This suggests that the model is able to accurately identify conjunctions in the input sentences.

Overall, the high accuracy and performance of the BiLSTM model demonstrate the effectiveness of using deep learning techniques for POS tagging.