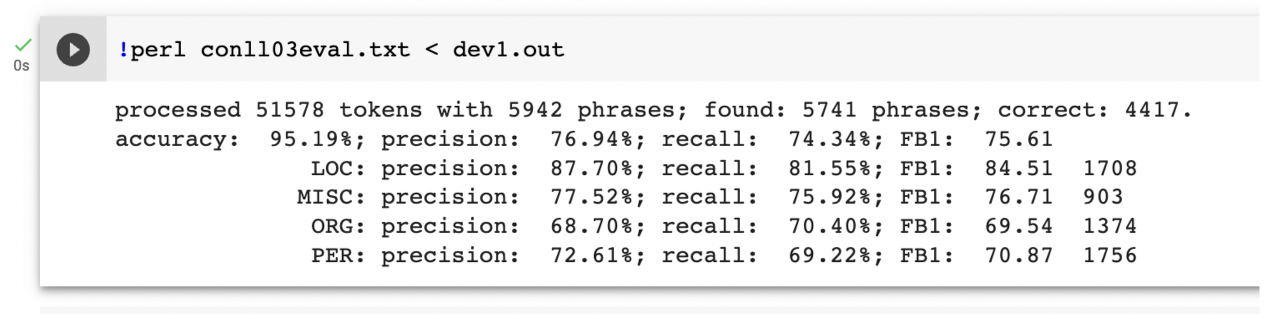
Task 1:

1.What are precision, recall and F1 scores on dev data?
Sol: Below are the values for model on dev data -

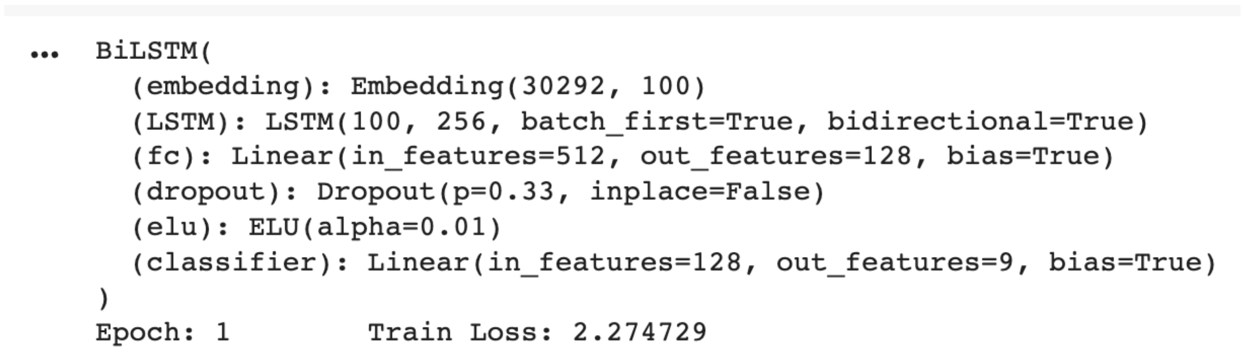


```
!perl conll03eval.txt < dev1.out

processed 51578 tokens with 5942 phrases; found: 5741 phrases; correct: 4417.
accuracy:  95.19%; precision:  76.94%; recall:  74.34%; FB1:  75.61
              LOC: precision:  87.70%; recall:  81.55%; FB1:  84.51  1708
             MISC: precision:  77.52%; recall:  75.92%; FB1:  76.71  903
              ORG: precision:  68.70%; recall:  70.40%; FB1:  69.54  1374
              PER: precision:  72.61%; recall:  69.22%; FB1:  70.87  1756
```

Solution description:

1. Architecture



```
BiLSTM(
  (embedding): Embedding(30292, 100)
  (LSTM): LSTM(100, 256, batch_first=True, bidirectional=True)
  (fc): Linear(in_features=512, out_features=128, bias=True)
  (dropout): Dropout(p=0.33, inplace=False)
  (elu): ELU(alpha=0.01)
  (classifier): Linear(in_features=128, out_features=9, bias=True)
)
Epoch: 1        Train Loss: 2.274729
```

As described in the HW4 pdf, there is only 1 lstm layer. We have an embedding layer initialized with vocabulary length and embedding dim=100.
Here, the weights in the embedding matrix are initialized by the model randomly. We have padding and unknown tokens defined but we do not initialize any value in the embedding matrix and let the model initialize it.
We have added a hidden layer of dim 256. These hidden layers are needed for LSTM to distinguish between padded and non-padded positions in a vector representation. We are also adding the dropout layer before our linear layer. Finally we have an ELU layer and linear layer as a classifier. We are also using pack_padded_sequence and pad_packed_sequence.

2. Hyper-params
We have the following hyper-parameters and their values -

Model	Unk threshold	Learning rate	Epochs	LR decay	Moment um

Task1	-	0.1	84	-	0.9

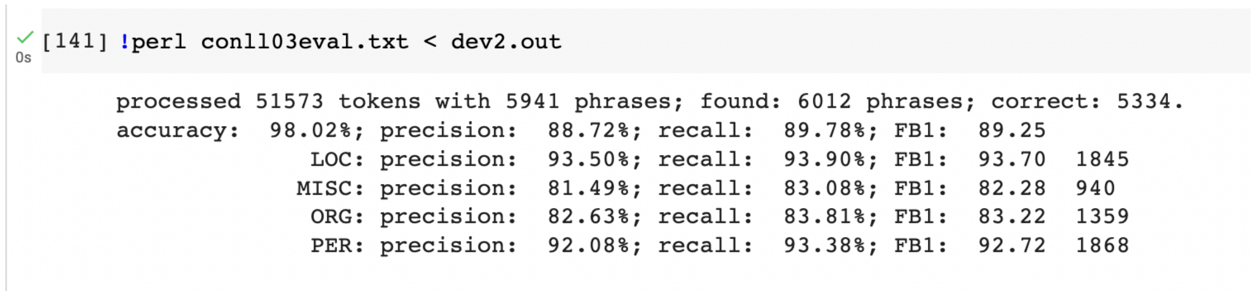
Class weights - `tensor([4.0900, 1.0000, 3.0000, 2.7300, 4.1500, 2.3600, 2.3000, 2.4600, 3.0200])`

3. Model training

We are following below steps for model training -
A. Create sentence list from Training, Dev and Test files
B. We have created a vocab dictionary from train data, that assigns an index to every word
C. We have created a tag dictionary to map every NER tag with an index
D. We then convert every word sentence in train data to its index (one-hot encoding) and each word is represented by a size of 100.
E. We are giving inputs to the model in batches and for every batch, we compute loss for every word using CrossEntropyLoss along with SGD optimiser.
F. Once we have predictions from our model, we are mapping the input, ytrue and ypred back from indices to original words & tags using our vocab dictionary.

Task 2:

1.What are precision, recall and F1 scores on dev data?
Sol: Below are the values for model on dev data -

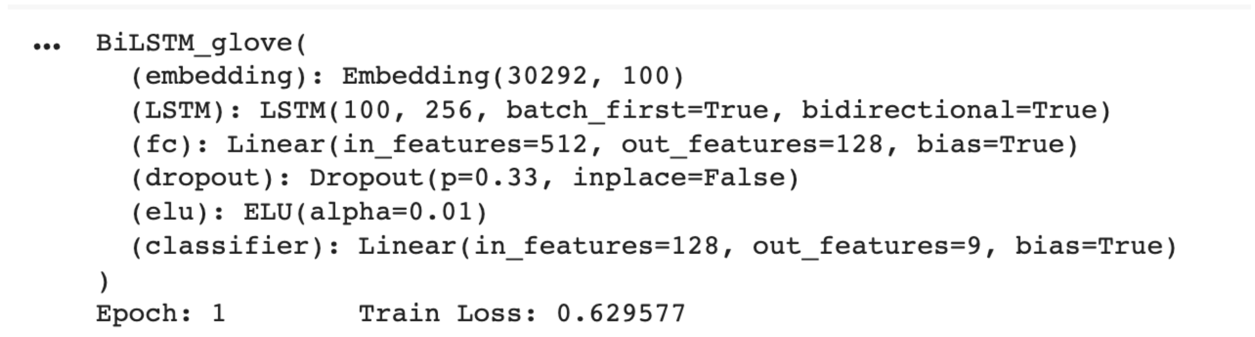


```
✓ [141] !perl conll03eval.txt < dev2.out
0s

      processed 51573 tokens with 5941 phrases; found: 6012 phrases; correct: 5334.
      accuracy:  98.02%; precision:  88.72%; recall:  89.78%; FB1:  89.25
                    LOC: precision:  93.50%; recall:  93.90%; FB1:  93.70  1845
                   MISC: precision:  81.49%; recall:  83.08%; FB1:  82.28  940
                    ORG: precision:  82.63%; recall:  83.81%; FB1:  83.22  1359
                    PER: precision:  92.08%; recall:  93.38%; FB1:  92.72  1868
```

Solution description:

1. Architecture



```
...  BiLSTM_glove(
       (embedding): Embedding(30292, 100)
       (LSTM): LSTM(100, 256, batch_first=True, bidirectional=True)
       (fc): Linear(in_features=512, out_features=128, bias=True)
       (dropout): Dropout(p=0.33, inplace=False)
       (elu): ELU(alpha=0.01)
       (classifier): Linear(in_features=128, out_features=9, bias=True)
     )
     Epoch: 1        Train Loss: 0.629577
```

As described in the HW4 pdf, there is only 1 lstm layer. We have an embedding layer initialized with vocabulary length and embedding dim=100. Here, the weights in the embedding matrix are initialized by using Glove vectors. We also initialized the matrix with values for padding and unknown tokens.

The value for padding in the embedding matrix is initialized with zeros.The value for the unknown token in the embedding matrix is calculated using the mean of rest of glove vectors.

We have added a hidden layer of dim 256. These hidden layers are needed for LSTM to distinguish between padded and non-padded positions in a vector representation. We are also adding the dropout layer before our linear layer. Finally we have an ELU layer and linear layer as a classifier.

2. Hyper-params

We have the following hyper-parameters and their values -

Model	Unk	Learning	Epochs	LR	Moment

	threshold	rate		decay	um
Task2 - Glove	-	0.1	50	Step size - 15 , gamma 0.9	0.9

Class weights - `tensor([4.0900, 1.0000, 3.0000, 2.7300, 4.1500, 2.3600, 2.3000, 2.4600, 3.0200])`
For the first few epochs,we are training model with a high learning rate. Once the model has learnt,we set this property criterion.requres_grad = True and decrease learning rate using StepLR

3. Model training

We are following below steps for model training -
A. Create sentence list from Training, Dev and Test files
B. We have created a vocab dictionary fromGlove vectors instead, that assigns an index to every word
C. We have created a tag dictionary to map every NER tag with an index
D. We then convert every word sentence in train data to its index (one-hot encoding) and each word is represented by a size of 100.
E. The word embeddings here are initialized using Glove vectors.
F.. We are giving inputs to the model in batches and for every batch, we compute loss for every word using CrossEntropyLoss along with SGD optimiser.
G. Once we have predictions from our model, we are mapping the input, ytrue and ypred back from indices to original words & tags using our vocab dictionary.

For word capitalisation issue:
We are checking if a word exists in Glove vocab, if yes we assign the embedding vector from Glove. If a word does not exist, the word is capital, then we make the word in lowercase and then assign an embedding vector from Glove.