CS728 Assignment 1 Report

Team Members:
Goda Nagakalyani - (214050010)

Dataset:

FIGER Data Split:

Train- 1248761 Test- 278
Dev- 9967

Implementation Details: Data Preprocessing:

For LSTM and BERT we have preprocessed data in the following manner. For a token having multiple tags in train/dev/test we have taken the tag with max length as a gold label.
The preprocessed data can be found in the folder /additional.

1.train_data_preprocessed.json 2.test_data_preprocessed.json 3.dev_data_preprocessed.json

For evaluation we have used Loose Macro and Loose Micro proposed in the dataset paper. We have attached the file evaluation_of_ner.py in /additional folder.

CRF:
Tried Implementation of CRF with sklearn_crfsuite, pytorch-crf, and Prof. Soumen's CRF code. Here is a screenshot of training with sklearn_crfuite. Some of the problems we faced while implementing these models are described in the later sections. We were not able to completely run it as it was taking a lot of time.

Here are the results we got after 1 epoch using sklearn_crfsuite Loose Micro:
Precision - 0.74
Recall - 0.86 F1 - 0.80
Loose Macro: Precision - 0.02 Recall - 0.03
F1 - 0.02

LSTM:
Model: Bi-directional LSTM
Loss function: categorical_crossentropy,
optimizer: Adam(0.001) with the corresponding learning rate Activation function: Softmax
Batch size: 128
Epochs: 200

We got the predictions for the test data using LSTM and saved them in the /output folder. Based on the predictions, we have computed the Precision, Recall and F1 scores based on Loose Macro and Loose Micro.
Loose Micro:
Precision - 0.8714517700552127 Recall - 0.6886710127816847
F1 - 0.7693542837481363

Loose Macro:

Precision - 0.9788847849210917 Recall - 0.9593296347478453
F1 - 0.96900856133242
We have an accuracy of 0.95. (6598/6944 matched tags)

BERT:

We have finetuned BERT for the task of NER. Model: 'distilbert-base-uncased'
Learning rate: 1e-4
Weight_decay: 1e-5
Batch size: 16 Epochs: 30

We got the predictions for the test data using BERT and saved them in the /output folder. Based on the predictions, we have computed the Precision, Recall and F1 scores based on Loose Macro and Loose Micro.

Loose Micro:
Precision - 0.7759344813103738 Recall - 0.7586879523638417
F1 - 0.7672143061070882

Loose Macro:
Precision - 0.9701906303139074 Recall - 0.967770489515988
F1 - 0.9689790487696978

We have an accuracy of 0.98 (6871/6944 matched tags)

Error Analysis:

One of the most common errors we get for LSTM/BERT are multilabel classification. Misclassification of labels from a long desc to short desc, i.e. /organization/educational_institution will be predicted as a /organization and /person/author will be predicted as /person.

These types of predictions which are not exactly correct but correct to some extent can be reconsidered to calculate the final accuracy.

OOVs can be handled properly with applying some rule based classification or some other technique to improve accuracy for those instances.

Challenges: CRF:

Having a large feature space dimension for each token (i.e. one hot representation) causes training the model on gpu challenging. In CPU it takes more than 12Hrs for each epoch. While trying with the GPU we got a memory exhausted error.

Tried using some libraries like sklearn_crfsuite and pytorch-crf. But each comes with their own disadvantages. Sklearn_crfsuite doesn't provide us with the capability to modify the loss function. Pytorch-crf doesn't store the weight vectors for emission probabilities and works only with transition probabilities.

BERT/LSTM:

System requirement is very large for training these models. At least 15GB of ram is required to run training. It was also taking a lot of epochs to minimize the loss, which makes the total time taken for training into 2-3 days.

Considered out of vocabulary words as another token <OOV>. Should have handled OOV in a more correct way like, applying rule based inference and so.

Takeaways/Findings:

In case of CRF even after 1 epoch micro accuracy comes to 86% because the model was mostly predicting "O" for all tokens and not able to distinguish between various classes. As we have got around 86% of data with "O" class the micro accuracy comes out to 86%. In this case macro should be considered for evaluating the accuracy of the model i.e. around 2% after 1 epoch.

For LSTM and BERT macro F1 is around and above 95% but micro F1 for the same is around 76%. This difference in the metric can be seen because of the highly imbalanced class supports. While the "O" class contains most of the training instances (i.e 86% of the data). When we calculate the F1 score using micro it uses the weighted F1 score of each class according to their class support. So, F1 micro gives a very high score because of the very high score we are getting for the "O" class (which is not desirable). We can consider Macro F1 as our performance metric by considering the prediction accuracy of each class as the same weight.

Conclusion:

From the above experiment we can infer that BERT performs better than LSTM for NER tasks. The LSTM also takes a lot of epochs (200) to converge while BERT can achieve its optimum pretty faster than LSTM with (30) epochs.