Slot Tagging for Natural Language Utterances

Sree Latha Chopparapu
UC Santa Cruz
schoppar@ucsc.edu

1 Introduction

Slot tagging for natural utterances is a phenomena of identifying contiguous spans of words in an utterance that correspond to certain parameters (i.e., slots) of a user request/query. In this problem we use the **IOB** labelling format for slot tagging. The ”B-” prefix indicates that the word is at the beginning of a slot, the ”I-” prefix indicates that the word is on the inside or at the end of a slot (occurs when slots contain at least 2 words). The ”O” label is assigned to words not belonging to any slot. In this task, We used a supervised approach for training the model.

1.1 Data

The dataset provided is generated based on film schema of Freebase knowledge graph. The file hw2_train, is a .csv file which contains the data for training. There are specifically 3 columns and 2313 rows. First column is ID which is the index of each row. The second column is ”UTTERANCES”, is a natural language text in English, for which the IOB slots were labelled. The third column is ”IOB SLOT TAGS”. *The table below illustrates an example of train data...*

show	credits	for	the	godfather
O	O	O	B-movie	I-movie

In the training data the UTTERANCES are the labels, which has a lot data imbalances such as the frequency of labels were varying too much For Example ”O” has a frequency of 1000+, where as ”B_location” has a frequency of 2, which is a huge difference. Figure 1 shows the frequency of all the IOB slots present in the data set along with their frequencies. As a process of pre-processing of text, the given data was analysed and found out the length differences in UTTERANCES and IOB SLOT TAGS. The data which has the length differences were dropped out of data file. There were few slot tags with I-movie instead of I_movie and replaced the particular string as necessary

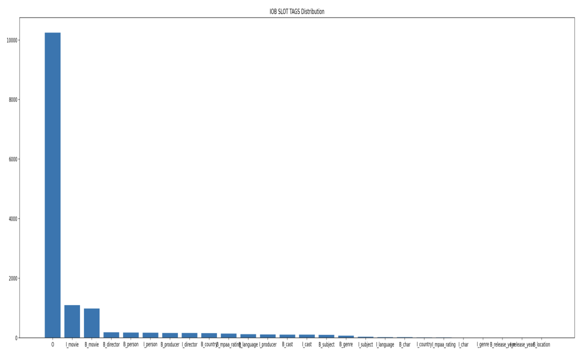

Figure 1: IOB slot tags distribution

Another file provided is hw2_test is a .csv file used for inference. The file consists of 2 columns and 982 rows of test data. First column is ID and second column is UTTERANCES, similar to the train data. Once the model is trained, we initiate the inference process which results in predictions often referred to as the logits. These inference results are structured in a .csv file with 2 columns, the first is for ID and the second column is for IOB SLOT TAGS obtained with respect to the input UTTERANCE during the inference process.

2 Model

2.1 Embeddings

The UTTERANCE data in the training data set has been tokenized using “spacy” open source libraries. The tokenized words are converted to a dictionary for encoding the key values of converted words. The same procedure has been followed for IOB labels for encoding. DataLoaders from PyTorch libraries were used for pipe-lining the data points.

2.2 Model Description

As in sequence labeling, we have to predict the output at each time step for every word occurring in the sequence, I have used an RNN (Recurrent Neural Network) for training the model. RNNs are a type of Neural Network where the output from the previous step is fed as input to the current step. As I have a very limited understanding on RNNs, I have kept the model quite simple, with an embedding layer of size obtained from vocab and a hidden size of 256.

The model was trained using the training data provided, keeping the hyper parameters initialized with some random but basic values. The training data used is split in the ratio 80:20 of training data and validation data. The hyper parameters used are hidden_size = 256, n_epochs = 20, learning_rate = 0.01, embedding_size = 256, n_label = 26

In the training process of neural network, it is very important to optimize the model parameters (weights) for minimizing the prediction error which is a highly complicated task. Certain algorithms are available for optimizing these weights and in this task Adam (Adaptive moment esti- mation)and SGD(Stochastic gradient descent) optimizers are used for optimizing the weights. While training the model, loss is calculated which the difference between predicted output and the ground truth. In the training process, we mainly focus on reducing the loss. For this task, I used, “CrossEntropyLoss” function from PyTorch framework, for the analysis of the loss during training.

3 Experiments

Data split involves splitting the dataset into multiple data sub sets such as training set, testing set and/or validation set. In this task, the data is split as training set and validation set in the ratio of 80:20 i.e. 80 percent of data as training data and 20 percent data to be validation data. While training the model the hyper parameters were tuned based on the gap between training loss and validation loss. I have conducted many experiments throughout the model training. The hyper parameters were constantly tuned to yield better results. In this task, I have tuned the learning rate to yield better results. I have kept the learning rate at 0.05 with a combination of ”Adam” and ”SGD” (individually...) and observed the erratic performance of the model. Then I decresed the learning rate to 0.01 and found out that the model is doing a little better compared to the previous step. Therefore, I have used an LR scheduler from *Optim* in PyTorch, to analyse the performance. At this point the model was performing good but as per my understanding it is overfitting, as the training accuracy is reaching around 90% but the test accuracy is only around 69%. From these experiments I have understood that the model I have trained is doing better when using an SGD optimizer than Adam (Surprising for me!). Table below illustrates the experiments.

EXP.NO	**LR**	**LOSS**	**ACC**
Exp1	0.01	15	72%
Exp2	0.05	7.4	70%
Exp3	LRSCH	15	84%

Table 1: Experiments with LR, Adam

EXP.NO	**LR**	**LOSS**	**ACC**
Exp1	0.01	2	94%
Exp2	0.05	3	93%
Exp3	LRSCH	3	92%

Table 2: Experiments with LR, SGD

4 Results

After numerous experiments with hyper parameter tuning and model dimensions etc, the Model trained was settling mostly on a testing accuracy of (Approx.)69 Percentage. Through out the training the complicated task was to choose a correct learning rate, which impacted the results drastically. Below plots shall illustrate the experiments graphically.

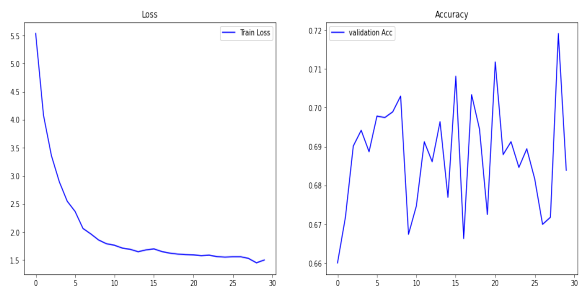

Figure 2: Train loss and Validation accuracy, Lr=0.01, optimizer-Adam

5 References

References

[1] https://neptune.ai/blog/recurrent-neural-network-guide

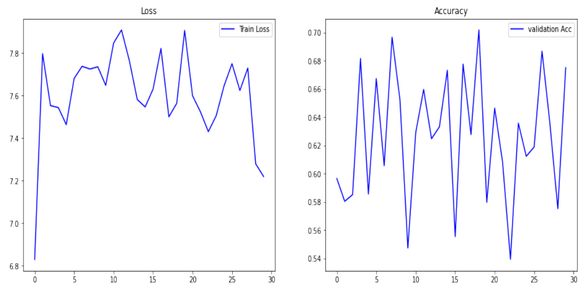

Figure 3: Train loss and Validation accuracy, Lr=0.05, optimizer-Adam

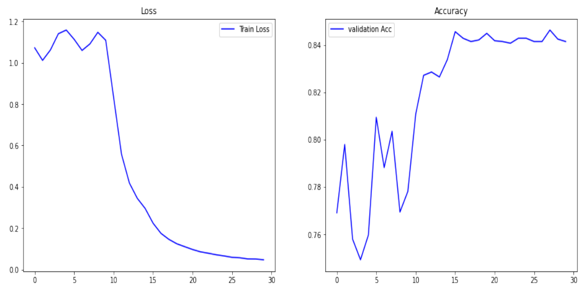

Figure 4: Train loss and Validation accuracy, LR_scheduler, optimizer-Adam

[2] https://gangaksankar.medium.com/lstm-model-for-ner-tagging-7c2018c51ece

[3] Gupta, R., Rastogi, A., Hakkani-Tur, D. (2018). An efficient approach to encoding context for spoken language understanding. arXiv preprint arXiv:1807.00267.

[4] https://www.dominodatalab.com/blog/named-entity-recognition-ner-challenges-and-model

[5] https://d2l.ai/chapter_recurrentneuralnetworks/rnn-scratch.html

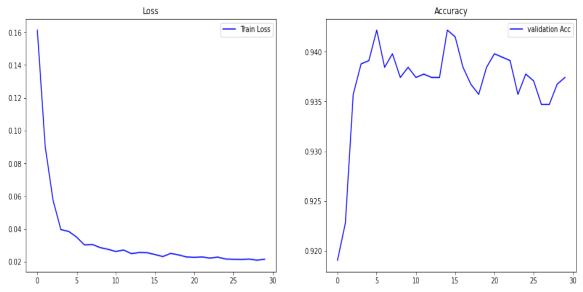

Figure 5: Train loss and Validation accuracy, LR=0.05, optimizer-SGD

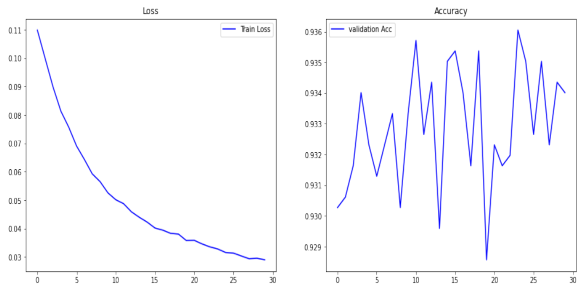

Figure 6: Train loss and Validation accuracy, LR=0.01r, optimizer-SGD

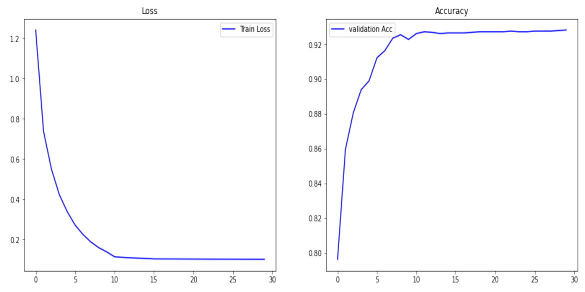

Figure 7: Train loss and Validation accuracy, LR_scheduler, optimizer-SGD