Deep Learning : Project 3

Prajjwal Mishra
SR NO. 17101

May 13, 2020

1 Logistics Regression with tf-idf

- **Pre-Processing**
  For text pre-processing all characters changed to lower case,joined text and hypothesis with newline characters to give a single sentence,lemmatizer has been done on the words to convert them to their root word and similarly snowball stemming has also been applied , after that and tokens are made from the sentences.
- In Logistics Regression with tf-idf used to create vectors out of the text , first create a vocabulary of 15000 words and then create vectors to represent the sentences and fit tfIdf vectorizer.
- **linear model**
  - validation Loss : 0.4044
  - validation Accuracy :0.595509
  - Test Accuracy : 0.591001

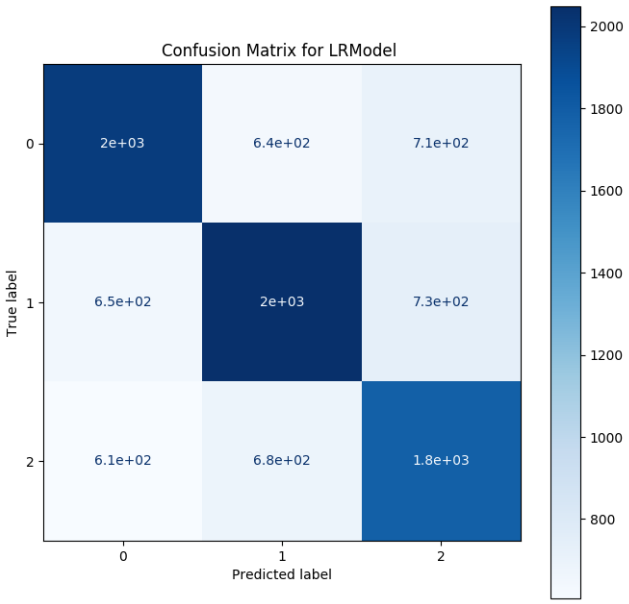

Figure 1: Confusion Matrix for Linear Logistics Regression

- **SGD_Classifier**
  - validation Loss : 0.44625
  - validation Accuracy : 0.553749
  - Test Accuracy : 0.54387

2 Feature Extraction for NN Models

Glove Vectors

- After Cleaning the dataset in which all the text with '-' unknown gold_label was discarded
- I used Spacy library to convert the whole sentence into a vector based from the model "en-core-web-sm" which is a pre-trained model . it converts a word into its embedding vector of length 96 and if passed a sentence it averages all the vectors of words and gives a mean vector .

3 Models

FeedForward Neural Network :

- MODEL ARCHITECTURE

```
Net(
  (fc1): Linear(in_features=192, out_features=100, bias=True)
  (fc2): Linear(in_features=100, out_features=60, bias=True)
  (fc3): Linear(in_features=60, out_features=40, bias=True)
  (fc4): Linear(in_features=40, out_features=3, bias=True)
)
```

- **Results**
  - TrainLoss 0.817304
  - ValidationLoss 0.864925
  - Accuracy on validation Set : 60 %
  - Test Accuracy : 60.199

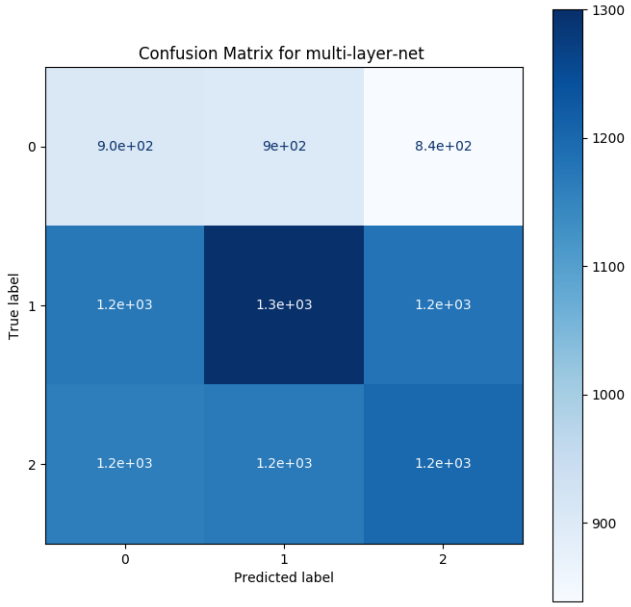

Figure 2: Confusion Matrix for FeedForward Neural Network

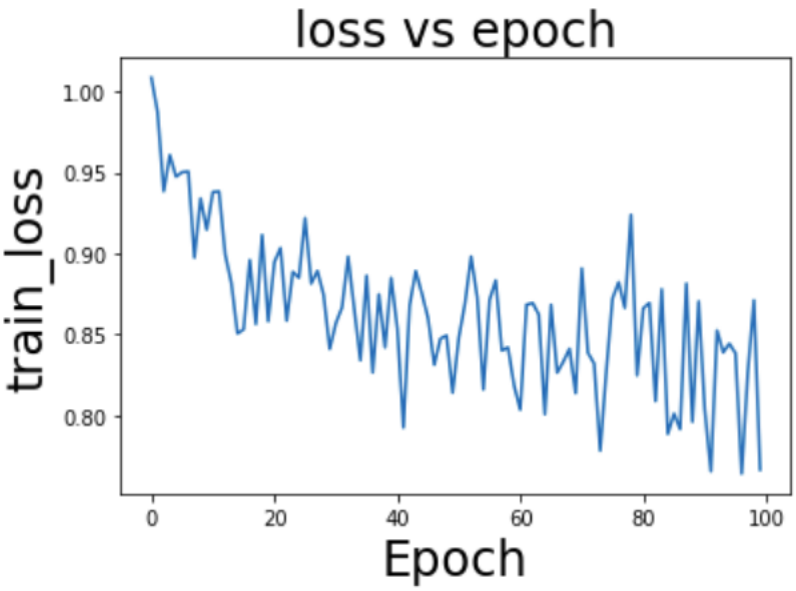

Figure 3: train loss vs Epoch

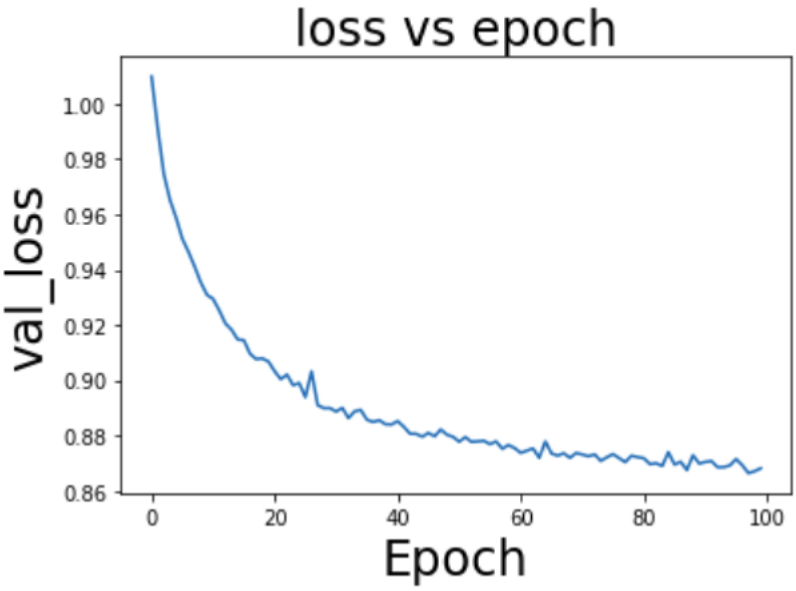

Figure 4: val loss vs epoch

- I used Log_softmax function , 100 epochs for training , Adam optimizer and a learning rate of "0.0001" chose this after running with different learning rates.

Recurrent Neural Network (LSTM)

- MODEL ARCHITECTURE

```
LSTMModel(
  (lstm): LSTM(48, 100, num_layers=2, batch_first=True)
  (fc): Linear(in_features=100, out_features=3, bias=True)
)
```

- In this model i used input dimension 48 half of the embedding vector of a sentence and sequence dimension 4 completing a whole of 192 , dimension of hidden layer is 100 and there are two such hidden layers with optimizer Adam and learning rate of).0001.
- **Results**
  - Train Loss 0.817304
  - Validation Loss: 0.8061335
  - Accuracy on validation Set : 60%
  - Test Accuracy : 60

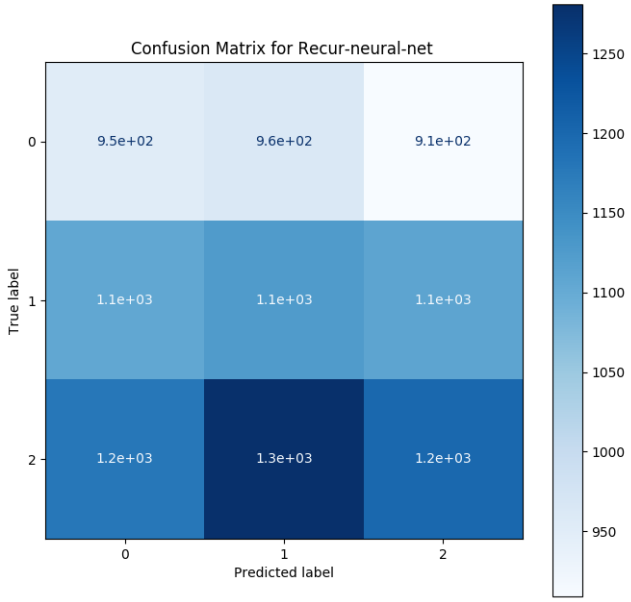

Figure 5: Confusion Matrix for Recurrent neural net

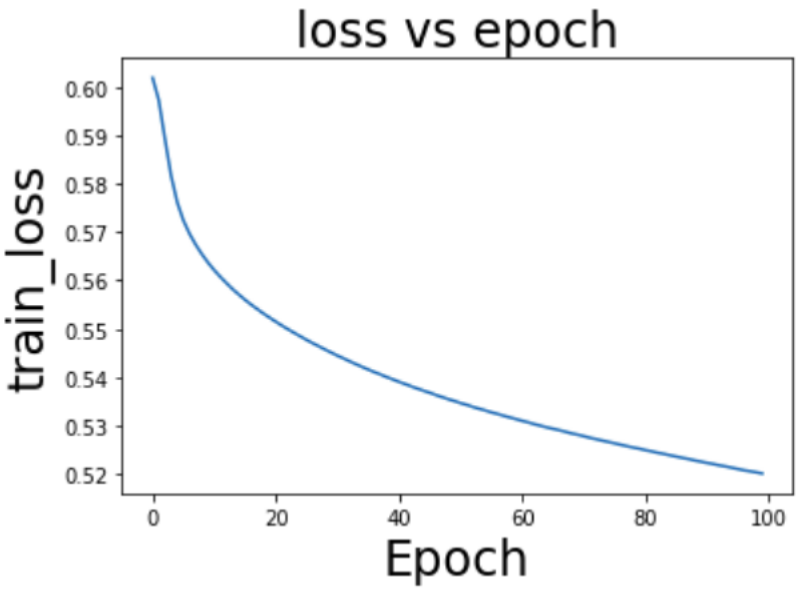

Figure 6: train loss vs epochs

Bert Model [Fine Tuned]

- It is a model trained by "Google" , the model is SOTA for many of the nlp tasks classification is one the tasks , here our task is related to classification task .
- Bert is based on encoder-decode model it has a vocabulary lookup table which maps

words to its hashed values , bert has its own tokenizer which converts the word/sentence's to tokens and with the help of lookup table a vector is assigned to each token .

- we can use the pretrained bert model and train it on our dataset according to our task to give better output .
- The model has 12 stacked encoders which takes input in a fixed format optimizer used is adamW that is adam with weight decay , learning rate was 2e-5.
- number of epochs was only one as was taking too long to train the model even with gpu's.
- **Results**[1]
  - Train Loss 0.43
  - Validation Loss: 0.36
  - Accuracy on validation Set : 87%
  - Test Accuracy : 89%

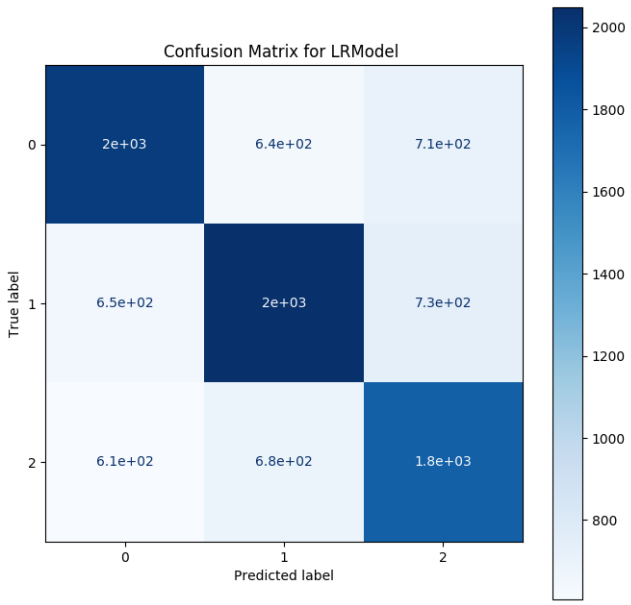

Figure 7: Confusion Matrix for Bert Model

[1]I'm not enclosing the loss graph because ran only 1 epoch on Bert Model as it takes too much time in getting trained