E0 270:MACHINE LEARNING

ML Assignment 2

Submitted By:
Name: Chetan Pant
SR no : 18071

Question 2:

In this question we build a autoencoder for learning unsupervised representations of a text.For encoder we use Bidirectional LSTM and for decoder we use unidirectional LSTM.For the model we use Pytorch.

1.**Processing the dataset:**

We tokenize every given english sentence in the training,validation and testing set. For tokenization of English sentences, Regexp tokenizer of NLTK is used. This tokenizer is selected for tokenization because while tokenizing it takes care of Apostrophe symbol. For example, if the sentence is "I'll be there", then Regexp tokenizer will tokenize it as ["I'll", 'be', 'there'] whereas word_tokenizer() of NLTK will tokenize it as ['I', "'ll", 'be', 'there'].

2.**Building the dictionary:**

To build the english vocabulary, first a list of tokens is generated using the tokenizer functions and then checked if the token already exists in the dictionary. If the token is not present in the dictionary then it is assigned an index and added to the dictionary. . One dictionary maps the word to its corresponding index (word2index) and another maps the index to the corresponding word (index2word).

3.Defining the Model Architecture:

For encoder a bidirectional LSTM is used, where the forward RNN goes over the embedded sentence from left to right and the backward RNN goes over the

embedded sentence from right to left. Due to the bidirectional nature of the encoder, we get two context vectors, one corresponding to each RNN. However, since the decoder used is unidirectional LSTM, so these context vectors are concatenated together through a linear layer and then the tanh activation function is applied and then passed to decoder.

```
seq2seq(
  (encoder): Encoder(
    (dropout): Dropout(p=0.5, inplace=False)
    (embedding): Embedding(26630, 100)
    (lstm): LSTM(100, 100, bidirectional=True)
    (linear_hidden): Linear(in_features=200, out_features=100, bias=True)
    (linear_cell): Linear(in_features=200, out_features=100, bias=True)
  )
  (decoder): Decoder(
    (dropout): Dropout(p=0.5, inplace=False)
    (embedding): Embedding(26630, 100)
    (lstm): LSTM(100, 100, dropout=0.5)
    (linear_decoder): Linear(in_features=100, out_features=26630, bias=True)
  )
)
```

4.**Preparing data for training the Model:** To create batches of same length, I calculated the maximum length of sentence in a batch and stored this value in a dictionary with key as batch_id. After obtaining the maximum length for each batch, "<pad>" token was appended to the sentences whose length was less than the maximum length of sentence in that batch. After that, Dataloader is used to create batches of the required batch size. Each batch will have sentences of same length.

5.Training the Model:

Model is trained for 35 epochs and training loss is calculated and also validation loss.We save the model after every 5 epochs.

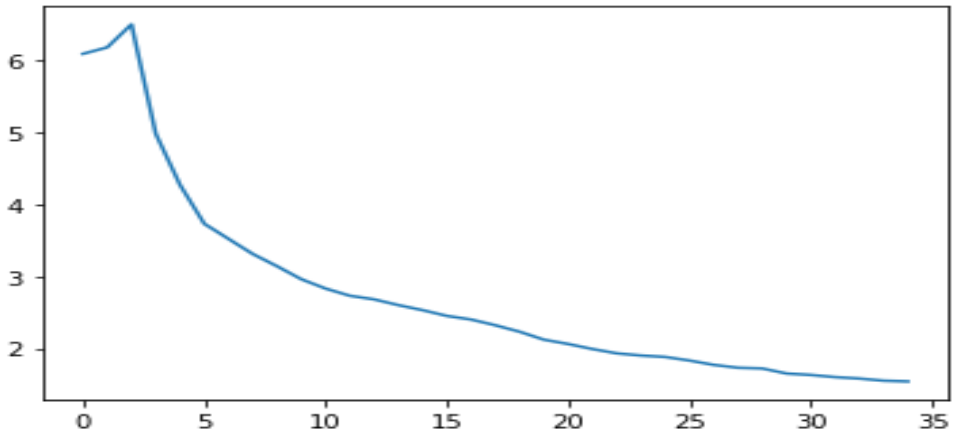
Note :AdamW optimizer and Cross Entropy Loss Function are used for computation of loss and to update the parameters of the model.

6.Testing the Model (Loss and BLEU Score):

We load the test data and preprocess this same as train .We test our model on the testing set .For performance metrics we use Construction loss and BLEU score.

Observations:

a.) Graph validation loss per epoch:



b.) Construction loss on testing set: 1.502873

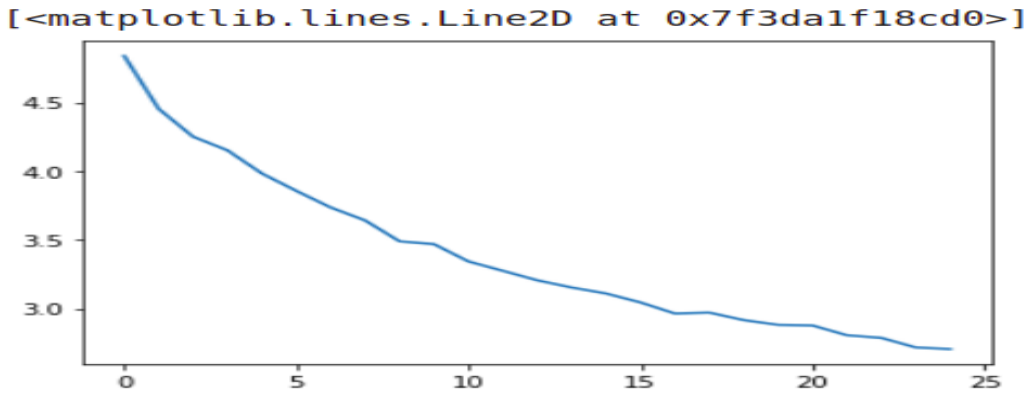
BLEU score : 0.590

c.) Effect of Teacher Forcing:

Here we only used teacher forcing value =1.0 and train the model for 25 epoch only.we can also evaluate on other teacher forcing values also.

With teacher forcing our model converges faster than without teacher forcing.

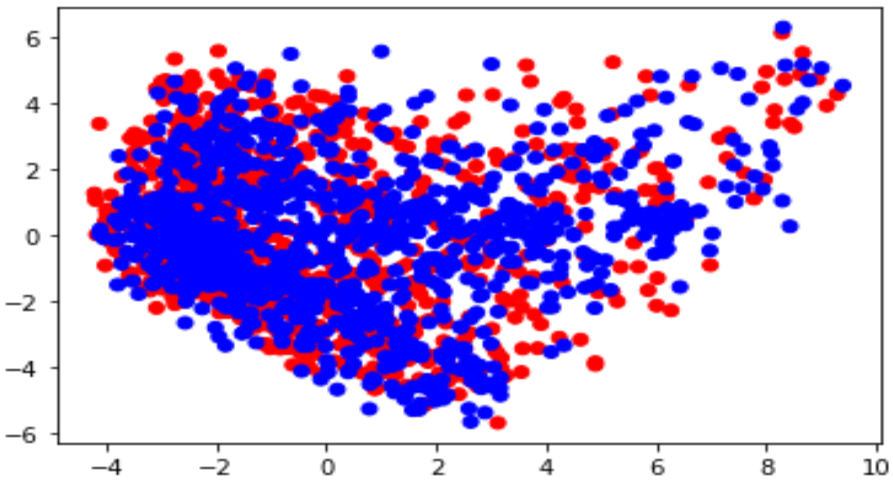
Plot for Validation loss per epoch (teacher forcing =1.0)



Construction loss on testing data: 2.87352

d.) dimensionality reduction and visualize it using a 2-D scatter plot along with their labels.(using PCA)

Plot:



e.) logistic regression classifier on the representations:

Classification accuracy on training data (Combined pos and neg) :76.45%

Classification accuracy on testing data (combined pos and neg) :72%

f.) LSTM based text classifier:

Model Architecture:

```
TextClassifier(
  (dropout): Dropout(p=0.5, inplace=False)
  (embedding): Embedding(26630, 100)
  (lstm): LSTM(100, 100, bidirectional=True)
  (linear_hidden): Linear(in_features=200, out_features=2, bias=True)
  (softmax): Softmax(dim=2)
)
```

Accuracy on training data: 98.6%

Accuracy on testing data :82.5%