ASSIGNMENT-2

Task 1

I have implemented the Part of Speech Tagging considering the following-

1. Corpus used – Pen Treebank section in the NLTK data
2. Vectors are trained along with the network with the help of RNN(Recurrent Neural Network) where I have used GRU
3. Implementation has been done using GRU(Gated Recurrent Unit) with Keras .

A POS tagging is the grammatical category of words where we can classify the given words into different tag-sets.

Input – The model will take in a sequence of words in a sentence.

Output – The corresponding POS tags for each word.

For Example – If input sequence is [fan, bag, he, she, in…], then the model should classify the sequence with the POS tags which can be – [NN, NN, VB, VB, IN…]

The Steps that I have followed for this implementation can be describes as follows:

1. The code starts with the importing the libraries required.
2. After this, the data has been downloaded from NLTK tree bank corpus . And two files have been created for this namely treebank_sents.txt and treebank_poss.txt which contains the input sentence data set and the corresponding pos tag lables for the words of the sentences respectively.
3. Next the data is parsed to find the maximum word frequency to build a look-up vocab dictionary for the words and for the tags. Also the total number of sentences in the training corpus has been found out. This function also finds out the total number of unique words. Ultimately, we will get the total number of unique tags from the corpus(45), the total number of sentences from the training corpus and the most frequent words.
4. Each row in the corpus is denoted as a sequence of the word indices. And the output will be the corresponding POS Tag indices.
5. Now, The datasets are fed to the network . The input sequence has been converted into a word ID sequence. And the build_tensor function is used to read the data from the two files and converts them into input and output tensors which can be further fed to the network.
6. The next step is to split the data-set into 80-20 train-test split.

7. The Model used in this approach can further be explained as follows:
   RNN (Recurrent Neural Network):

A Recurrent Neural Network is a class of artificial neural network where connections between the units from a directed graph are fed into a directed graph along a sequence. In RNN, if we assume a network containing RNN neurons, then each neuron will perform the same operation on every element of the sequence. There are two variants of the RNN – LSTM (Long short term memory (LSTM) and gated recurrent (GRU).

In the given research paper, Schmidt has implemented the Pos tagging using a 3-layer perceptron network (MLP network) connected by directed weights. The processing units are arranged vertically in several layers. The bottom layer is the input layer , top layer is the output layer and any layers between the input and the output layer are called the hidden layers. In this, the network learns by adapting the weights of the connections between the units until the correct output is predicted. The backpropagation using the gradient descent has been used here.

Approach:

Here , GRU , a variant of the LSTM has been used. GRU basically has 2 gates – an update gate and the reset gate. The update gate tells how much of the previous memory to keep and the reset gate tells how to combine the new input with the previous memory. The GRUs are faster to train as compared to other networks.

The input to the network, is a tensor of the words. This is then sent to the embedding layer which converts the word into a dense vector. The output tensor from this layer is the fed to the GRU layer. And then the output of this is finally passed through the activation function , Softmax which gives us a final output tensor. The maximum of each of the columns of the tensor gives us the index of each of the predicted POS tags for the word at that position. The model is trained for the given number of epochs.

Advantages-

1. We are not doing any feature engineering here, the Model learns the features by itself .
2. The word embedding are fed to the model with the help of which model accomplishes the above step during the training itself.

Accuracy-

The accuracy for this comes out to be 91% and the validation also as 91% using two epochs, hidden size to be 60 , Embed Size as 90 and Batch size to be 32.

The model can be improved if we train the model for more datasets, the accuracy can be improved to some extent. Here around 3000 samples are used. And also it can improved by varying the parameters and dimensions.

<u>Task 2: Dialogue Act Recognition</u>

The following has been considered for task-2-

1. Corpus used – NPS chat in the NLTK data
2. Vectors are trained along with the network with the help of RNN(Recurrent Neural Network) where I have used Bidirectional LSTM in Keras.
3. *Reference: [Antonio_Gulli,_Sujit_Pal]_Deep_Learning_With_Kera(b-ok.org)*

Input – The model will take in a sequence of words in a sentence.

Output – The corresponding sentiment value (statement, rejection, goodbye, emotion, etc)

The Steps that I have followed for this implementation can be describes as follows:

1. The code starts with the importing the libraries required.
2. After this, the data has been downloaded from NLTK nps_chat corpus. And two files have been created for this namely data.txt and test.txt. Data.txt(Training corpus) which contains the input sentence data set and the corresponding dialogue annotations for the sentences.
3. Each row in the corpus is denoted as a sequence of the word indices where the indices are ordered by most frequent to the least frequent words in the training set. And the output will be the corresponding dialogue.
4. The input sentences are converted to word index sequences. And the labels that need to be fed are also converted into a vector.
5. Now, the datasets are fed to the network.
6. The next step is to split the data-set into 80-20 train-test split.

7. The Model used in this approach can further be explained as follows:
8. **RNN (Recurrent Neural Network): (Bidrectional LSTM)**

A Recurrent Neural Network is a class of artificial neural network where connections between the units from a directed graph are fed into a directed graph along a sequence. In RNN, if we assume a network containing RNN neurons, then each neuron will perform the same operation on every element of the sequence. There are two variants of the RNN – Bidirectional LSTM (Long short term memory (LSTM) and gated recurrent (GRU).

<u>Approach:</u>

<u>Bidirectional LSTM</u> - It contains two hidden layers of opposite directions to the same output. The output layer gets the information from both the past and future states. So, it is sentiment analysis of the past words of the sentence can easily be considered for the prediction of the next words of the sentences. This increases the performance by keeping the results of the sequence of the words in consideration. It involves duplicating the first recurrent layer in the network so that here are now two layers simultaneously

The input to the network, is a tensor of the words. This is then sent to the embedding layer which converts the word into a dense vector. The output tensor from this layer is the fed to the bidirectional LSTM layer. And then the output of this is finally passed through the activation function, Softmax which gives us a final output tensor and is compiled using the categorical cross entropy. The maximum of each of the columns of the tensor gives us the index of each of the predicted dialogues for the given sentences. The model is trained for the given number of epochs.

Advantages-

1. We are not doing any feature engineering here, the Model learns the feature
2. The word embedding is fed to the model with the help of which model accomplishes the above step during the training itself.
3. Bidirectional RNN can be trained using the

Accuracy-

The accuracy for this comes out to be 94% with 7 epochs, hidden Layer size to be 100, Embed Size as 128, and Batch size to be 32.

Ideas for improvement-

1. We can improve the model to get better accuracy using GRU instead of Bidirectional LSTM.
2. Also, if we train it on a larger data set , that can also improve the performance and the accuracy of the model.
3. It can further be improved by varying the parameters and dimensions used in the present model.

References-

1.https://machinelearningmastery.com/develop-bidirectional-lstm-sequence-classification-python-keras/

2. [Antonio_Gulli,_Sujit_Pal]_Deep_Learning_With_Kera(b-ok.org).pdf

3. https://en.wikipedia.org/wiki/Bidirectional_recurrent_neural_networks

4. https://en.wikipedia.org/wiki/Recurrent_neural_network

5. Discussion Credits: pmorparia@iu.edu, ukoul@iu.edu , gkbandep@iu.edu