ASSIGNMENT-3

Task 1: TEXT CLASSIFICATION USING CNN

The following has been considered for task-2-

1. Corpus used – IMDB Movie Corpus
2. Vectors are trained along with the network with the help of CNN (Convolutional Neural Network) in Keras.

Input – The model will take in a sequence of words in a sentence (The movie Content)

Output – The corresponding sentiment value (Negative/Positive)

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. Corpus Generation- The corpus has been generated with the help of the positive and negative sentiments of test and train data in the file trainfile.txt.
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 positive or negative sentiment.
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 Architecture and implementation of the model

    1. The Architecture of the Model involves adding one Embedding layer with the parameters as vocab_size(This is the minimum of the total frequency of the words and the total number of features), embedding size being 32 and input length to be the maximum sentence length.
    2. The next layer added is the Convolution 1D layer with parameters as filters (32), kernel size being 3, and padding being 'same'(which returns a filter map of the same size and automatically appends zeros) and activation function used here is 'Relu'
    3. The next layer is for Max Pooling. This reduces the size of filter maps by applying max filter to the non-overlapping sub regions. The max pooling layer with pooling size = 2 (2*@ max filters) reduces the total number of parameters in the filter map by a factor of 4.
    4. The next layer is Flatten layer. Flattening is used to convert the input activations into a 1D array.

5. After this, two Dense layers have been added. Dense layer is a fully connected layer of neurons which is applied on the 1D input and gives a 1D output. The first Dense layer uses ‘relu’ activation function and the second activation layer uses ‘sigmoid’ activation function and 2 classes.
6. Finally, the model is compiled using the parameters –
   Loss function being ‘binary_crossentropy, optimizer being ‘adam’ and metrics function to measure accuracy.
7. After this we print the model summary.

Approach:

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 Convolutional 1D layer. And then the output of this is finally passed through Max pooling layer and two dense layers as described above which gives us a final output tensor and is compiled using the binary_crossentropy. The maximum of each of the columns of the tensor gives us the index of each of the predicted sentiments for the Movie content. The model is trained for 2 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. The training in CNN is very fast and gives better performance and accuracy.

Accuracy-

The accuracy for this comes out to be 89% with 2 epochs.

Task 2: TEXT CLASSIFICATION USING RNN

The following has been considered for task-2-

1. Corpus used – IMDB Movie Corpus
2. Vectors are trained along with the network with the help of CNN (Convolutional Neural Network) in Keras.

Input – The model will take in a sequence of words in a sentence (The movie Content)

Output – The corresponding sentiment value (Negative/Positive)

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. Corpus Generation- The corpus has been generated with the help of the positive and negative sentiments of test and train data in the file trainfile.txt.
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.

The Model used in this approach can further be explained as follows:

RNN (Recurrent Neural Network): (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 – LSTM (Long short term memory (LSTM))

Approach:

LSTM - LSTM is a variant of RNN which can learn long term dependencies. It implements recurrence by using the hidden state from the previous time step and the current input in a tanh layer. LSTM is used to construct and train a many-to-one RNN. The network takes the movie content sentence (Sequence of words) and outputs a sentiment value (positive or negative).

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 sentiments for the given movie content 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.

Accuracy-

The accuracy for this comes out to be 87% with 4 epochs, hidden Layer size to be 100, Embed Size as 128, and Batch size to be 32.

Comparison between RNN and CNN for Text Classification-

Training Complexity- RNN takes more time for training the model even on GPU and CNN takes less time to train the model.

Performance Results – CNN gives better accuracy as compared to RNN for the task of text classification.

References-

1. Data Set reference - Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts. Learning word vectors for sentiment analysis. In Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies, pages 142–150,Portland, Oregon, USA, June 2011. Association for Computational Linguistics

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. https://www.kaggle.com/danielelton/keras-cnn-with-explanation

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