ML in CL - Assignment 3

Question 1 b) Sentence Classification using CNN for movie review data.

Data: The dataset was taken from the github repo - https://github.com/AcademiaSinicaNLPLab/sentiment_dataset
This repo gives a pkl file “MR.pkl” which consists of sentences and their class (either 0 or 1). 0 represents a negative review and 1 represents a positive review. Below is a sample output from the “MR.pkl” file. There are 10662 text samples in total.

	label	sentence	split
62	0	the cumulative effect of watching this 65 minu...	train
10190	1	rarely has skin looked as beautiful , desirabl...	train
5385	1	behind the snow games and lovable siberian hus...	train
6632	1	the weight of the piece , the unerring profess...	train
6822	1	based on a devilishly witty script by heather ...	train

Data Pre-processing: The text data has to be provided in a numerical format for any neural network. To achieve this, “Keras Tokenizer” was used. The Tokenizer takes the whole corpus as input and keeps only the top “N” words to convert text into numeric value. With each word having a numeric value, each sentence can be represented as a sequence of numbers. If a sentence has a word which is not in the top “N” words of the Tokenizer, that word is ignored from the sentence representation. Basically, the intuition is that the top “N” words are enough to represent every sentence.

Vectorization: A CNN or RNN expects the input to have a fixed size. To convert every sentence to a same sized sequence, zero-padding is done such that 0`s are added at the beginning and the sequence length is maintained to the value specified as a parameter. In the case for movie review, I have set the maximum sequence length as 52. With this approach, every review is represented as a sequence of 52 numbers possibly beginning with 0`s.
With a fixed size input sequence, CNN or RNN also expects input to be a 2D matrix. The reviews have to be now converted into a 2D matrix. For this task, I used Glove embeddings for each word. Now each word in the sentence is represented as a 100 dimensional vector since I chose the 100 dimensional Glove embeddings.
These two processes now make the input sequence (movie review) as a (52,100) matrix.

Model: The model is a single layer 1D- convolutional network with Max-Pooling.

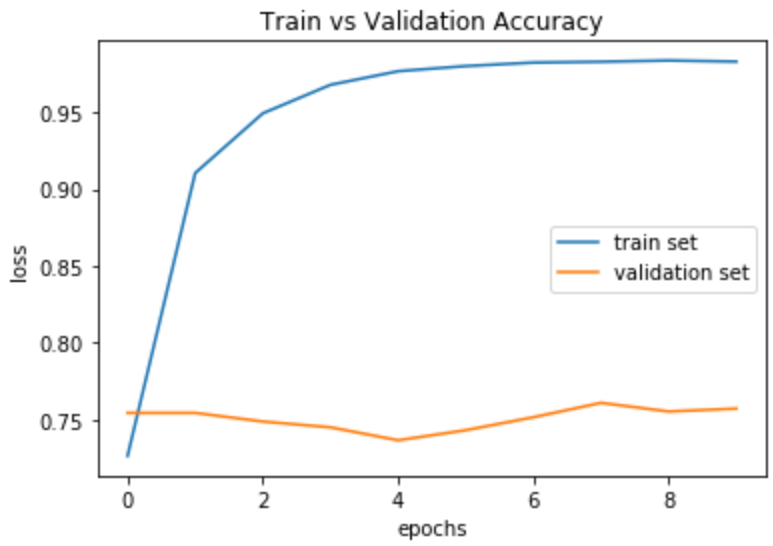
Layer (type)	Output Shape	Param #
input_6 (InputLayer)	(None, 52)	0
embedding_1 (Embedding)	(None, 52, 100)	1875900
conv1d_10 (Conv1D)	(None, 48, 128)	64128
max_pooling1d_10 (MaxPooling)	(None, 9, 128)	0
flatten_6 (Flatten)	(None, 1152)	0
dense_11 (Dense)	(None, 100)	115300
dense_12 (Dense)	(None, 2)	202

Total params: 2,055,530
Trainable params: 2,055,530
Non-trainable params: 0

The input is a sequence of size 52. Each of the 52 word has an embedding of size 100 which is fed into the embedding layer. A 1D convolution is applied to this layer. The convolution layer has 128 filters with each filter of size 5. This convolution reduces the dimension of the input from (52,100) to (48,128). A max pooling is applied over this matrix to get a reduced matrix of size (9,128). The size of max pooling is 5. It reduces the input by a factor of 5. This reduced matrix is now fed to two fully connected layer with the last fully connected layer having softmax as the activation to get the output probabilities.

Result: The model required very few epochs to get a very good training accuracy (98%). This is because of the usage of pre-trained Glove embeddings. But the validation accuracy was just around 76% for 1067 samples. There is a significant difference. This shows that the model overfits easily and it requires some regularization to avoid overfitting. Below is a graph of training and validation accuracy which clearly explains overfitting.

Code can be found here (My github repository) - CNN_Movie_Review.ipynb

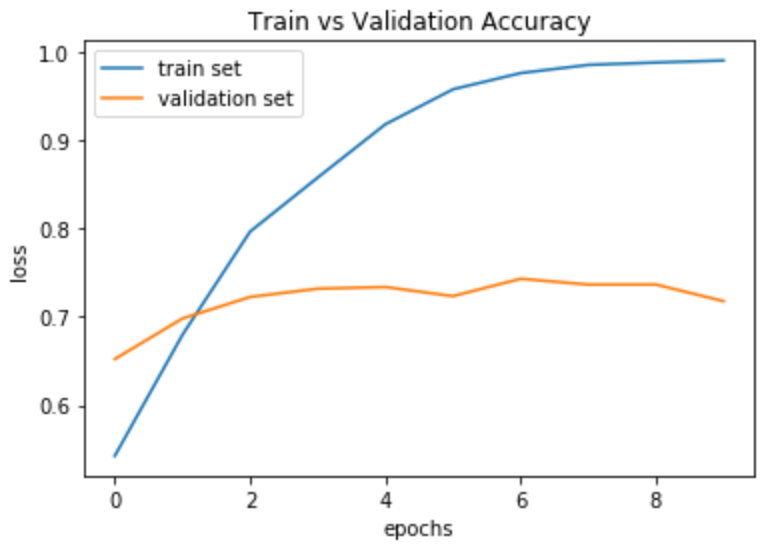


Question 1 c) Sentence Classification using CNN for movie review data along with NLP features.

This task involved addition of NLP features to the network. I added a probability distribution of POS tag for every word to the embedding. The dimension of embedding layer changed from 100 to 145 to incorporate the distribution for all the POS tags. The probability distribution of POS tags for every word is calculated as follows:

A matrix of size of N*45 is initialized to zeros. Then the nltk POS tagger is run for all words in sentences and the words' POS tag count is updated. Let's say the word "bank" appears as a noun twice and as a verb once. So the matrix now is [NN…….VB] =[3,.....1] . This matrix is now normalized by taking the sum of every row and dividing each element by its row sum. So the final distribution becomes [NN…...VB] = [0.75,.....0.25]. This vector is appended to the pre-trained embeddings and the same CNN as mentioned in 1b is trained for this. Below is the training and validation accuracy graph. The graph indicates that there is a slight decrease in the validation accuracy for 10 epochs in total. It also shows that validation accuracy is almost constant after few epochs. It does not make much sense to run for more epochs even though there are more parameters here compared to the previous CNN version.

Code can be found here (My github repository) - CNN_Lexical_Prob.ipynb



Question 2) Implement RNN for the same task - Sentence Classification

For this task, the Data, Data Pre-processing method, Vectorization method remains the same. The only change is in the model architecture which is a variant of RNN called LSTM.

Model: The model architecture is shown below

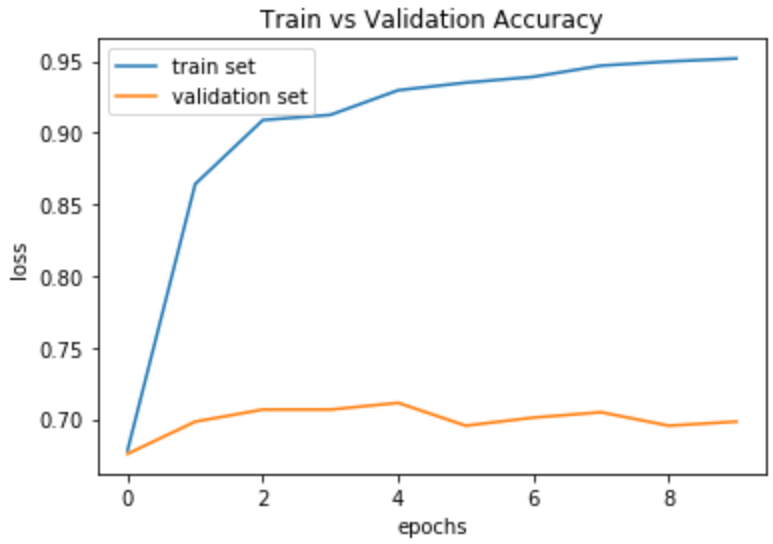
Layer (type)	Output Shape	Param #
input_12 (InputLayer)	(None, 51)	0
embedding_1 (Embedding)	(None, 51, 100)	1875900
lstm_26 (LSTM)	(None, 51, 50)	30200
lstm_27 (LSTM)	(None, 51, 40)	14560
dropout_16 (Dropout)	(None, 51, 40)	0
lstm_28 (LSTM)	(None, 51, 20)	4880
dropout_17 (Dropout)	(None, 51, 20)	0
lstm_29 (LSTM)	(None, 51, 10)	1240

```
global_max_pooling1d_12 (Glo (None, 10)                0
_________________________________________________________________
dropout_18 (Dropout)         (None, 10)                0
_________________________________________________________________
dense_12 (Dense)             (None, 2)                 22
=================================================================
Total params: 1,926,802
Trainable params: 1,926,802
Non-trainable params: 0
```

There are 4 LSTM cells in total with Dropout for regularization between lstm_27 and lstm_ 28, lstm_28 and lstm_29, lstm_29 and Dense layer. These dropouts are added to avoid overfitting. The first parameter for every LSTM cell is the number of hidden units which itself is a hyper-parameter. Return_sequences = True means that the output of the LSTM cell is passed onto the next LSTM cell.

The Global Max pooling in 1d is commonly used for text classification to reduce the data from 2d to 1d. In our case, the output from the last LSTM was (51,10). Applying a Global Max Pooling after this gives a max value from each of the 10 vectors of size 51. This reduces the dimension to just a 10 dimension vector. The term “Global” is a variant of the Max Pooling layer except that the pool size is equal to the input size. In our case, pooling size is 51. The training and validation accuracy graph is shown below.

Code can be found here (My github repository) - [RNN_Movie_Review.ipynb]



Comparison of network architectures: The CNN network clearly outperformed the RNN network both in terms of training and validation accuracy. In fact, CNN converged to a higher training and validation accuracy much quicker.
Reasoning - For a text classification task like this one, the “order of text is irrelevant”. All we care here is whether the text expresses a positive or a negative opinion towards a movie. A CNN captures N-gram based features and does a repeated convolution on this. The last layers somewhat represent the higher representation of a part of a text (summary of a part of text maybe). On the contrary, an RNN gives a lot of weightage to the order of words and processes every word one step at a time instead of a whole set of words. It is trying to hard to figure out how each word contributes to the whole sentence (either positively or negatively).