Prerak Mehta

MSDS 458 Artificial Intelligence and Deep Learning

Third Research Assignment

Abstract:

In this assignment we are working with a text classification problem using the AG's news topic classification dataset. AG is a collection of more than 1 million news articles. News articles have been gathered from more than 2000 news sources by ComeToMyHead in more than 1 year of activity. ComeToMyHead is an academic news search engine which has been running since July, 2004. The dataset is provided by the academic community for research purposes in data mining (clustering, classification, etc), information retrieval (ranking, search, etc), xml, data compression, data streaming, and any other non-commercial activity. The AG's news topic classification dataset is constructed by choosing 4 largest classes (World, Sports, Business, and Sci/Tech) from the original corpus. Each class contains 30,000 training samples and 1,900 testing samples. The total number of training samples is 120,000 and testing 7,600. In this assignment, the comparison will be made among the performances of various RNN models built differently. The difference between the RNN models particularly lie in terms of number of LSTM layers and hidden nodes per layer.

Introduction:

This research is being conducted to address a hypothetical management problem - classify random news articles documents in their respective categories. RNN (Recurrent Neural Network) models are constructed in different ways to explore what impacts the classification accuracy the most for this particular research problem. This research assignment serves as a hands-on practical experience with not only designing, training and assessing a recurrent neural

network but also understanding whether employing more LSTM layers and hidden nodes per layer positively impacts the accuracy of the text (article) classification model.

Literature Review:

A lot of research has been done in this area however there isn't a singleton company that just focuses on a product which builds neural network models to classify texts. Social media platforms such as Facebook, Instagram, Twitter, WhatsApp, LinkedIn, etc. use AI and DL techniques to assign metadata tags from the backend to news, statuses, articles, etc. shared by their users in order to classify and categorize their content and use it efficiently to recommend their users familiar content. Models used by these platforms are extremely complex and unique to their purpose. However this model focuses on building a fundamental foundation that compares performances of various types of RNN models that are known to solve the automated text classification problems. It mainly focuses on the impact LSTM layers and hidden nodes per layer have on the performance of the RNN models.

Methods:

Firstly, after loading the AG news dataset, get all the words in the documents by using the encoder to get the indices associated with each token and then translating the indices to tokens. But first we need to get the "unpadded" news articles so that we can get their length. Then before performing some EDA on the dataset, some information needs to be retrieved about the dataset in order to know what we're dealing with. The total of training and test data is 127,600 news articles evenly distributed among the 4 categories. We used the tf.keras.layers.experimental.preprocessing.TextVectorization layer to transform each news article into a "list" of non-negative integers representing the tokens in the news article. For the purpose of training our models each such "encoding" will have a fixed length corresponding to the news

article(s) with the most tokens. Shorter articles will be right-padded with zeros in the encoding. Also to speed up the training process, we will set max_tokens = 1000 so that words not in the vocabulary set of the top 1000 most common tokens are encoded as 1. But first we set max_tokens = None (which is the default value) in order to get the vocabulary size of the corpus.There are 95976 vocabulary words in the corpus. There are 95976 vocabulary words in the corpus. Then encode the news articles using the top 1000 most common words in the corpus. In particular, 0 is used for padding, 1 for the unknown words, 2 for the common word, i.e. 'the', etc. We later determine the number of non-vocabulary words in each news articles (denoted by 1s in the encoding). Then we preprocess Shuffle Data for Training and Create Batches of (text, label) pairs. The text encoder basically standardizes each sample (usually lowercasing + punctuation stripping), splits each sample into substrings (usually words), recombines substrings into tokens (usually ngrams), indexs tokens (associate a unique int value with each token), and transforms each sample using this index, either into a vector of ints or a dense float vector.Once the vocabulary is set, the layer can encode text into indices. The tensors of indices are 0-padded to the longest sequence in the batch. The model we use in this assignment is tf.keras.Sequential. The first layer is the encoder, which converts the text to a sequence of token indices. After the encoder is an embedding layer. An embedding layer stores one vector per word. When called, it converts the sequences of word indices to sequences of vectors. These vectors are trainable. After training (on enough data), words with similar meanings often have similar vectors.
This index-lookup is much more efficient than the equivalent operation of passing a one-hot encoded vector through a tf.keras.layers.Dense layer. A recurrent neural network (RNN) processes sequence input by iterating through the elements. RNNs pass the outputs from one timestep to their input on the next timestep. The tf.keras.layers.Bidirectional wrapper can also be

used with an RNN layer. This propagates the input forward and backwards through the RNN layer and then concatenates the final output. The main advantage to a bidirectional RNN is that the signal from the beginning of the input doesn't need to be processed all the way through every timestep to affect the output. The 6 models that are tested in this assignment are:

1) 1 LSTM layer using text encoder of top 1000 vocabulary words.
2) 1 LSTM layer with more hidden nodes using text encoder of top 1000 vocabulary words.
3) 2 LSTM layers using text encoder of top 1000 vocabulary words.
4) 3 LSTM layers with more hidden nodes using text encoder of top 1000 vocabulary words.
5) 1 LSTM layer with more nodes and text encoder using all the vocabulary words.
6) 2 LSTM layer and text encoder using all the vocabulary words

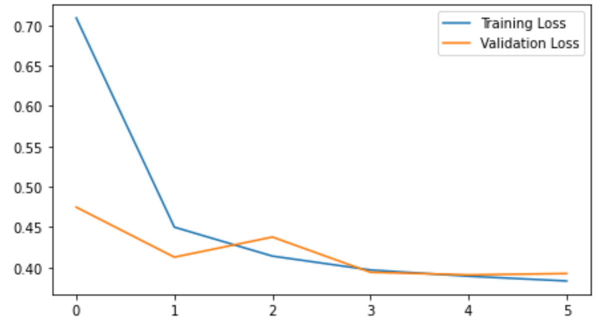
.**Results:**

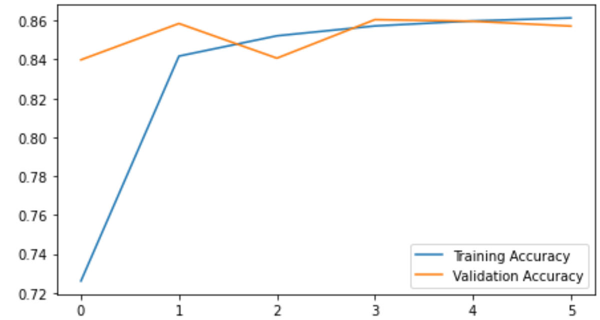
We create a dataframe to stack up all the model performances for comparison and also use Matplotlib to create 2 plots - displaying the training and validation loss and training and validation accuracy for each training epoch side by side.

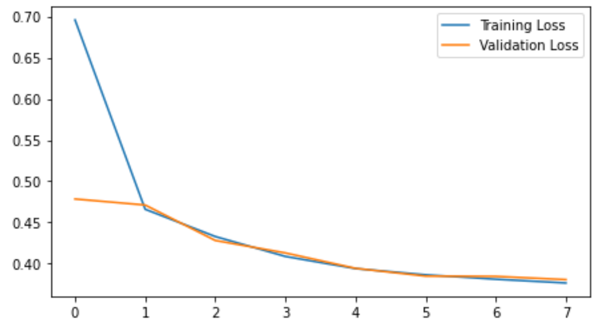
The performances (accuracies and losses) of all the six models in a dataframe are as below:

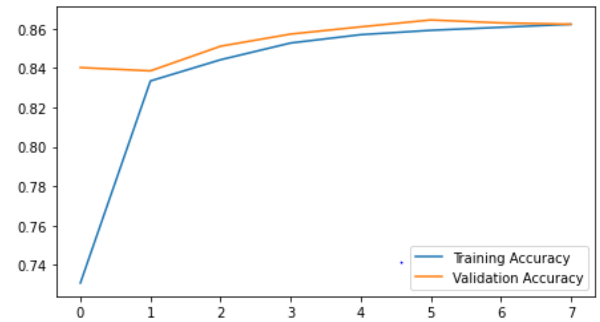
	Model Number	LSTM Layers	Encoder Vocab	Max Nodes	Processing Time (seconds)	Training Set Accuracy	Training Loss	Validation Set Accuracy	Validation Loss	Test Set Accuracy	Test Loss
0	1	1	1000	64	3085.671875	0.864149	0.374952	0.857167	0.392325	0.851579	0.408124
1	2	1	1000	128	10339.625000	0.868035	0.363146	0.862167	0.380758	0.854342	0.397909
2	3	2	1000	64	9344.656250	0.871842	0.356145	0.864500	0.381850	0.851974	0.403298
3	4	3	10000	128	23723.859375	0.866281	0.370586	0.857000	0.395507	0.852105	0.411843
4	5	1	None	128	12765.000000	0.962465	0.114098	0.881833	0.393015	0.880789	0.394459
5	6	2	None	64	7460.718750	0.963842	0.121154	0.889833	0.363615	0.887105	0.369371

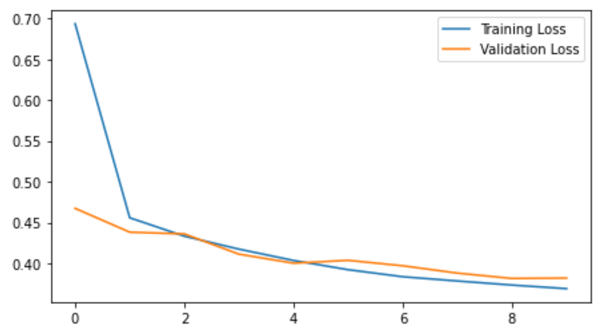
The plots for all the six models respectively are as below:

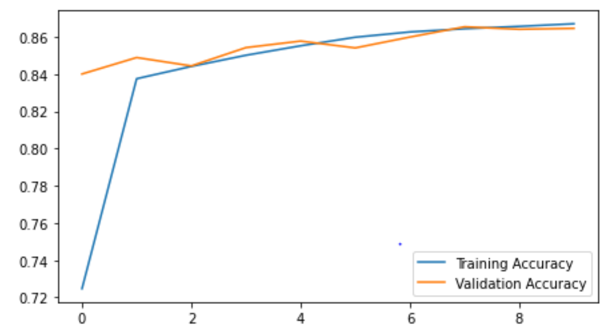


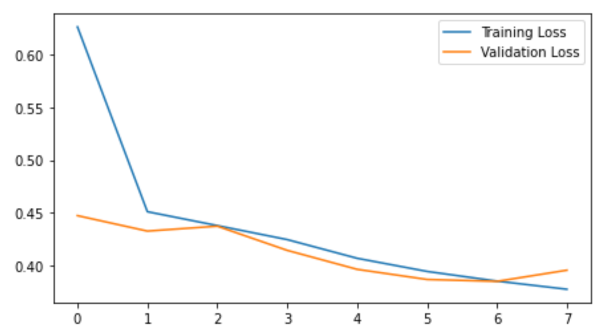


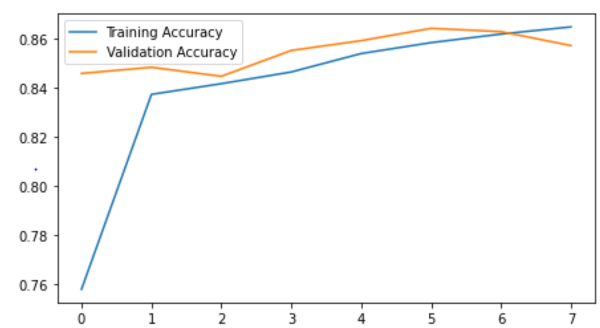


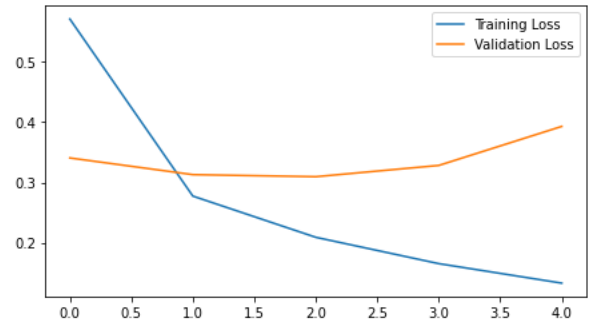


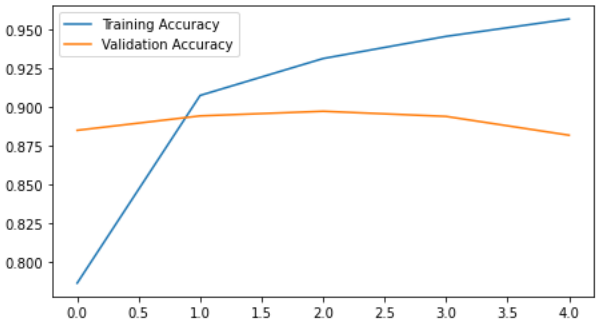


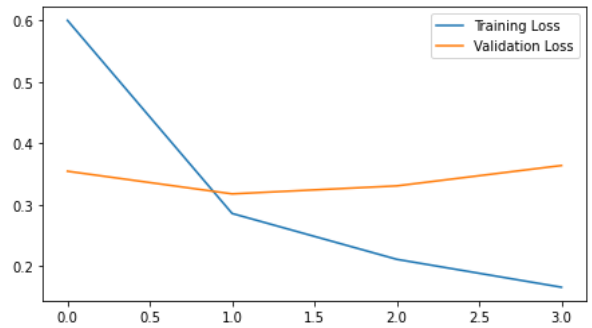


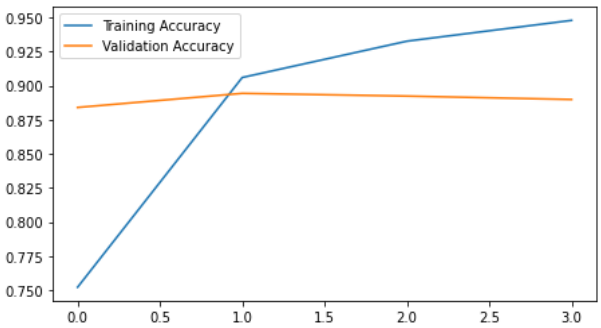












Conclusion:

In conclusion, the research assignment gave a lot of insights about the impact of extra LSTM layers and nodes in a Recurrent Neural Network, We learned about some advantages of the RNN but the main disadvantage of a bidirectional RNN is that you can't efficiently stream predictions as words are being added to the end. After the RNN has converted the sequence to a single vector the two layers.Dense do some final processing, and convert from this vector representation to a single logit as the classification output. All in all this research assignment gave a decent head start to building a conversational agent or chatbot to assist customer support representatives.