MSDS 458 Fall 2021 Assignment 3
Husein Adenwala

1. Abstract

The purpose of this research was to review various deep learning architectures for NLP and various text pre-processing techniques to determine the best possible model and data pre-processing steps that can be used to automate document text classification in an organization. I worked with the AG News Subset data from TensorFlow, which consists of 127600 news articles categorized in 4 classes. I experimented with various vocabulary setup and hyperparameters for a Simple RNN and with LTSM layers (both unidirectional and bidirectional) and worked with a 1-dimensional CNN to determine the best algorithm and vocabulary setup for this data. Results from my experiment suggest that LSTM and CNN models perform better than RNN models; however, the CNN model had the best test accuracy and took the least amount of training time.

2. Introduction

The bulk of data generated today is unstructured data; therefore, it's important that organizations find ways to manage and analyze it. Organizations receive a lot of unstructured text data in documents they need to classify so that it is easier to manage. With a manual approach, staff would need to sort through each text and assign a label or category to it individually. The problem is that manual classification can be time-consuming, error-prone, and cost-prohibitive.

Looking at the success of chat bots that use NLP for text classification, a company has consulted me to develop a natural language processing (NLP) classifier to automate the classification of their text documents into one of several predefined categories.

3. Literature review

Deep learning-based models have outperformed various classical machine-learning-based approaches in various text classification tasks, including sentiment analysis, news categorization, question answering, and natural language inference (Minaee et al, 2021).

RNNs (or LSTMs) have been established as advanced approaches for natural language processing. CNNs are generally used in computer vision; however, they've recently been applied to various NLP tasks with promising results.

However, attention-based models have outperformed simpler RNN based model (Glassi et al, 2021). Attention based models such as BERT and XLNet currently have the best performance for NLP tasks. However, attention-based models significantly increase the parameters of the model which make them computationally expensive to train.

4. Methods

Research design and modeling methods

For this research, Recurring Neural network, Long Short-Term Memory (LSTM) and Convolutional Neural Net models are analyzed and compared with varying architectures for each.

Recurring Neural networks are feedforward artificial neural networks that can deal with sequential data and can be trained to hold the knowledge about the past. Long Short-Term Memory (LSTM) networks that

are a type of recurrent neural network capable of handling long-term dependencies in sequence prediction problems and Convolutional Neural Net are designed to learn spatial hierarchies of features. CNN can extract local and position-invariant features. On the other hand, an RNN-LSTM learns order dependencies in sequence.

I have experimented with several vocabulary sizes, output sequence length, layers of RNN, LSTM and CNN models and Regularization techniques.

All models use the ReLU activation function in the dense and convolution layers, which is one of the popular activation functions in neural networks because it is computationally efficient and fixes the problem of vanishing gradient. I used validation Accuracy as the metric to evaluate the performance. Finally, I used Sparse categorical cross entropy to measure the loss between labels and predictions and SoftMax activation function for the output layer.

Data preparation, exploration, visualization

The dataset used in this research is the AG's news topic classification dataset which is constructed by choosing the 4 largest classes from the original AG news corpus that contains more than 1 million news articles. Each class contains 30,000 training samples and 1,900 testing samples. The total number of training samples is 120,000 and has 7,600 testing samples. The training data was split to create a validation data set of 6000

The 4 classes in the dataset are **'World'**, **'Sports'**, **'Business'**, **'Sci/Tech'**

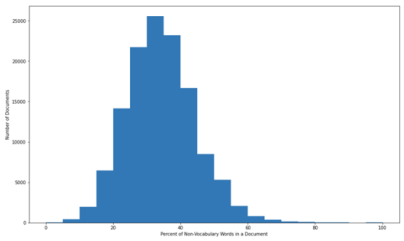
Some examples of news article in tabular form:

description	label
AMD #39;s new dual-core Opteron chip is designed mainly for corporate computing applications, including databases, Web services, and financial transactions.	3 (Sci/Tech)
Reuters - Major League Baseball\Monday announced a decision on the appeal filed by Chicago Cubs\pitcher Kerry Wood regarding a suspension stemming from an\incident earlier this season.	1 (Sports)
President Bush #39;s quot;revenue-neutral quot; tax reform needs losers to balance its winners, and people claiming the federal deduction for state and local taxes may be in administration planners #39; sights, news reports say.	2 (Business)
Britain will run out of leading scientists unless science education is improved, says Professor Colin Pillinger.	3 (Sci/Tech)
London, England (Sports Network) - England midfielder Steven Gerrard injured his groin late in Thursday #39;s training session, but is hopeful he will be ready for Saturday #39;s World Cup qualifier against Austria.	1 (Sports)
TOKYO - Sony Corp. is banking on the \$3 billion deal to acquire Hollywood studio Metro-Goldwyn-Mayer Inc...	0 (World)
Giant pandas may well prefer bamboo to laptops, but wireless technology is helping researchers in China in their efforts to protect the engandered animals living in the remote Wolong Nature Reserve.	3 (Sci/Tech)
VILNIUS, Lithuania - Lithuania #39;s main parties formed an alliance to try to keep a Russian-born tycoon and his populist promises out of the government in Sunday #39;s second round of parliamentary elections in this Baltic country.	0 (World)
Witnesses in the trial of a US soldier charged with abusing prisoners at Abu Ghraib have told the court that the CIA sometimes directed abuse and orders were received from military command to toughen interrogations.	0 (World)
Dan Olsen of Ponte Vedra Beach, Fla., shot a 7-under 65 Thursday to take a one-shot lead after two rounds of the PGA Tour qualifying tournament.	1 (Sports)

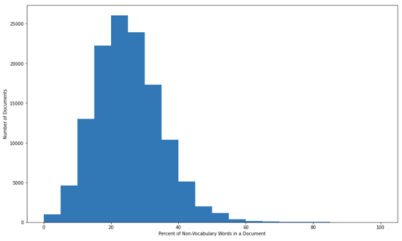
Preprocessing data:

There are 95976 unique words in the vocabulary of the total dataset and if we reduce the vocabulary to 3 distinct levels 1000, 2000 and 3000 we get the following out of vocabulary distribution.

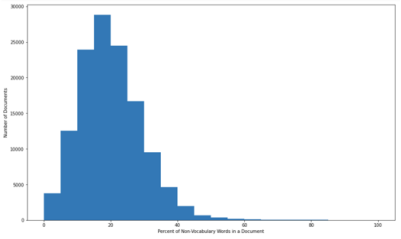
Vocabular size= 1000



Vocabular size= 2000



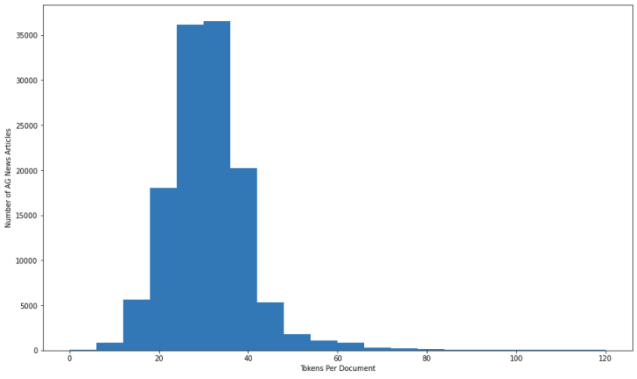
Vocabular size= 3000



```
"the" is in the vocabulary.
"dog" is *not* in the vocabulary.
"ran" is in the vocabulary.
"after" is in the vocabulary.
"a" is in the vocabulary.
"red" is in the vocabulary.
"ball" is in the vocabulary.
"as" is in the vocabulary.
"it" is in the vocabulary.
"rolled" is in the vocabulary.
"by" is in the vocabulary.
"the" is in the vocabulary.
"hat" is in the vocabulary.
"on" is in the vocabulary.
"the" is in the vocabulary.
"ground." is *not* in the vocabulary.
```

With vocabulary size 3000 we still loose key words in our documents as seen in this example at the left.

Output sequence length:



There are 123 unique words per news article, but the majority of news articles have between 40 to 60 unique words. This can be used to set the output sequence length for some experiments.

Analysis of most and least frequent words show that the least frequent words don't add a lot of meaning to the news article.

20 most frequent words in the Vocabulary:

```
['', '[UNK]', 'the', 'a', 'to', 'of', 'in', 'and', 'on', 'for',
 'that', '39s', 'with', 'its', 'as', 'at', 'is', 'said', 'by', 'it']
```

20 least frequent words in the Vocabulary:

```
['0133', '0125', '0121', '012', '011micron', '0119', '01112004',
 '011104', '01102004', '011', '0100', '008s', '007percent', '007',
 '005', '004', '0013', '000th', '000strong', '000660se'],
```

Implementation and programming

Model research was done using Python programming language and TensorFlow. TensorFlow is a free and open-source software library for machine learning and artificial intelligence and TensorFlow can be implemented in python by Keras. Keras is a deep learning API written in Python, running on top of the machine learning platform TensorFlow which is extremely user-friendly.

Train, validation and test data set was loaded using TensorFlow load function which loads data as a tensor object.

I used the sci-kit learn library which has functions to create a confusion matrix and report on precision, accuracy and F1 score.

Additionally, I used NumPy and Pandas library for data for EDA and finally, I also used seaborn package and matplotlib library to visualize the data and results.

5. Results

Experiments:

33 models were evaluated by conducting. See Appendix A to see the results of all the models.

The experiments were done by varying model architecture, vocabulary lengths, output sequence length and regulation methods.

Results Top 10 Models

Name	Type	Architecture	Vocabulary Size	Output Sequence len	Regularization	Train Accuracy	Valid Accuracy	Test Accuracy	Time
Model 3C	CNN	1 layer CON1d 64 Kernel, MaxPool1D, 3, 1 Dense layer 64 Nodes	Max	60	Early stopping + DropoutEarly stopping + Dropout	94.25	91.08	90.68	1 min 37s
Model 4D	LSTM	1 Layer LSTM _Bldirectional 128 Cells + 1 Dense Layer 128 Nodes	Max	60	Early stopping + Dropout	8.29	92.6	89.63	3 min 2 S
Model 4B	LSTM	1 Layer LSTM _Bldirectional 128 Cells + 1 Dense Layer 128 Nodes	3000	60	Early stopping + Dropout	86.07	89.38	88.86	12 min 38s
Model 3A	CNN	1 layer CON1d 64 Kernel, MaxPool1D, 3, 1 Dense layer 64 Nodes	3000	40	Early stopping	89.91	89.15	88.83	4 min 42 s
Model 1BBB	LSTM	2 Layer LSTM _Bldirectional 64 + 32 Cells + 1 Dense Layer 64 Nodes	3000	Defualt	Early stopping + Dropout	89.03	89.12	88.71	4 min 0 s
Model 1AAA	LSTM	1 Layer LSTM _Bldirectional 64 Cells + 1 Dense Layer 64 Nodes	3000	Defualt	Early stopping	90.02	88.9	88.64	4 min 27 s
Model 4C	LSTM	1 Layer LSTM _Bldirectional 64 + 32 Cells + 1 Dense Layer 128 Nodes	3000	40	Early stopping + Dropout	92.46	90.42	88.52	11 min 50 s
Model 1CCC	LSTM	1 Layer LSTM _Unidirectional 64 Cells + 1 Dense Layer 64 Nodes	3000	Defualt	Early stopping	90.19	88.97	88.44	3 min 24 s
Model 2AAA	RNN	1 Layer RNN _Bldirectional 64 Cells + 1 Dense Layer 64 Nodes	3000	Defualt	Early stopping	90.81	88.75	88.39	36 min 17 s
Model 4A	LSTM	1 Layer LSTM _Bldirectional 128 Cells + 1 Dense Layer 128 Nodes	1000	60	Early stopping + Dropout	87.38	87	88.36	5 min 4 s

Key Findings:

Vocabulary size and output sequence:

Higher vocabulary size provided better performance to the model. Limiting the output sequence length that covers roughly 68% of tokens in vocabulary provides good results.

Early Stopping and dropout:

Overfitting was observed with models that had multiple LSTM, RNN and Con1D layers and early stopping and dropout did reduce the overfitting. Simpler models did not exhibit any significant overfitting and did not require dropout.

LSTM Models:

I experimented with single layers and multi-LSTM bidirectional and unidirectional layer models. The bidirectional had a slight edge over unidirectional models but there was not any significant improvement in performance by adding more layers or cell in the LSTM model which made the model overfit and added a lot of parameters that increased training time. The improvement in performance comes from setting the vocabulary length and output sequence size which standardized the length of all the new articles.

The best LSTM model received test accuracy of 89.63%. The accuracy for all LSTM models ranged from 84.64% to 89.63%

RNN Models:

I also conducted experiments on Simple Recurrent Neural Networks. RNN models performed better than expected with the best test accuracy of 88.39 in Model 2AAA. Increasing the complexity of the model reduced its performance. The RNN model seemed to hit a performance ceiling and could not improve with additional hyperparameter tuning.

Compared to both our CNN and LSTM models, RNN models performed less well and test accuracy ranged from 81.74% to 88.39%.

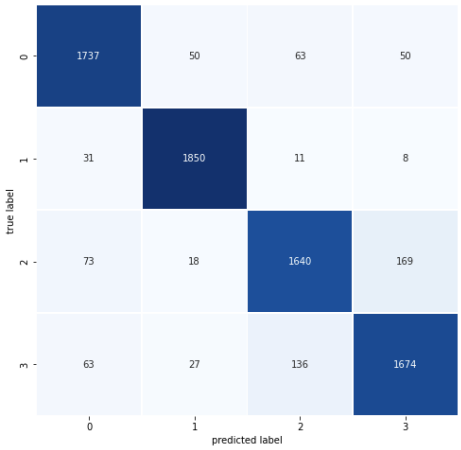
CNN Models:

I experimented with a variety of CNN models with different combinations of Conv/Max Pool Layer counts, number of filters, hidden layer counts, and hidden layer sizes. Of all of these models, the model with 1 Conv/Max Pool layer with 30% dropout had the highest test accuracy. Adding layers did not improve performance of the network but increasing the vocabulary size or setting vocabulary size to maximum and setting output sequence to cover majority new article length (60) provided the best performance.

The best LSTM model (Model 3C) received test accuracy of 90.68%. The accuracy for all LSTM models ranged from 88.75% to 90.68%

Best, Model 3C: 1 Layer Con1D with vocabulary set to maximum and output sequence to 60 tokens.

Name	Type	Architecture	Vocabulary Size	Output Sequence len	Regularization	Train Accuracy	Valid Accuracy	Test Accuracy	Time
Model 3C	CNN	1 layer CON1d 64 Kernel, MaxPool1D, 3, 1 Dense layer 64 Nodes	Max	60	Early stopping + DropoutEarly stopping + Dropout	94.25	91.08	90.68	1 min 37s




```
Classification Report
              precision    recall  f1-score   support

           0       0.91      0.91      0.91      1900
           1       0.95      0.97      0.96      1900
           2       0.89      0.86      0.87      1900
           3       0.88      0.88      0.88      1900

    accuracy                           0.91      7600
   macro avg       0.91      0.91      0.91      7600
weighted avg       0.91      0.91      0.91      7600

Accuracy Score: 0.9080263157894737
Root Mean Square Error: 0.5276561879022921
```

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
text_vectorization (TextVect (None, 60)                0
_________________________________________________________________
embedding (Embedding)        (None, 60, 64)            5811200
_________________________________________________________________
conv1d (Conv1D)              (None, 60, 64)            12352
_________________________________________________________________
max_pooling1d (MaxPooling1D) (None, 20, 64)            0
_________________________________________________________________
flatten (Flatten)            (None, 1280)              0
_________________________________________________________________
dense (Dense)                (None, 128)               163968
_________________________________________________________________
dense_1 (Dense)              (None, 4)                 516
=================================================================
Total params: 5,988,036
Trainable params: 5,988,036
Non-trainable params: 0
_________________________________________________________________
```

6. Conclusions

The results of the 33 experiments show that CNN performs better than LSTM and RNN. Compared to LSTM and RNN models, it was also computationally efficient. The intuition behind CNN performing better could be because the news articles are not too long where the maximum token count is 123; CNN

can extract one-dimensional spatial structure in the sequence of words and maybe it is also able to pick out invariant features in the different categories. The experiments also proved that adding layers in a CNN model or LSTM and RNN model did not improve accuracy significantly and deep models tended to overfit and take more time to train, especially for LSTM and RNN. CNN, LSTM and RNN models all had an inherent characteristic to overfit the data and needed regularization to reduce the variance.

Setting the optimum Vocabulary size and output sequence were key to improving performance of all the models and additional model performance gain could be accomplished if we remove the stop words from the corpus.

In conclusion, because of the highest test accuracy and relative computational efficiency, I recommend the CNN model 3C for document classification.

References:

Minaee, S., Kalchbrenner, N., Cambria, E., Nikzad, N., Chenaghlu, M.A., & Gao, J. (2021). Deep Learning--based Text Classification. *ACM Computing Surveys (CSUR), 54*, 1 - 40.

Y. Luan and S. Lin, "Research on Text Classification Based on CNN and LSTM," *2019 IEEE International Conference on Artificial Intelligence and Computer Applications (ICAICA)*, 2019, pp. 352-355, doi: 10.1109/ICAICA.2019.8873454.

Galassi, A., Lippi, M., & Torroni, P. (2021). Attention in Natural Language Processing. *IEEE Transactions on Neural Networks and Learning Systems, 32*, 4291-4308.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun: Deep Residual Learning for Image Recognition, Dec 2015, DOI: https://arxiv.org/abs/1512.03385

Frahcois, C. (2018). Deep Learning with Python. Shelter Island, New York: Manning Publications Co.

Geỉron, A. (2019). *Hands-on machine learning with Scikit-Learn, Keras and TensorFlow: concepts, tools, and techniques to build intelligent systems* (2nd ed.). O'Reilly.

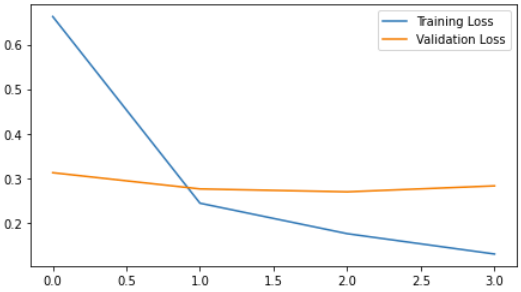
Appendix A

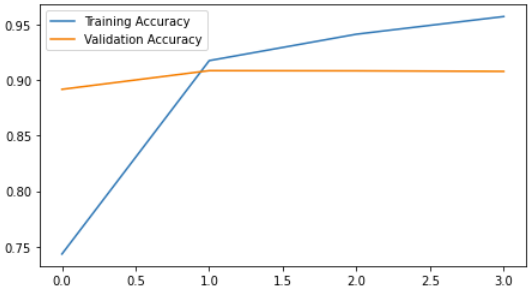
All Experiment Results

Name	Type	Architecture	Vocabulary Size	Output Sequence len	Regularization	Train Accuracy	Valid Accuracy	Test Accuracy	Time
Model 1A	LSTM	1 Layer LSTM _Bidirectional 64 Cells + 1 Dense Layer 64 Nodes	1000	Default	Early stopping	86.36	86.55	85.34	4 min 9 s
Model 1B	LSTM	2 Layer LSTM _Bidirectional 64 + 32 Cells + 1 Dense Layer 64 Nodes	1000	Default	Early stopping + Dropout	85.61	87.03	85.69	3 min 16 s
Model 1C	LSTM	1 Layer LSTM _Unidirectional 64 Cells + 1 Dense Layer 64 Nodes	1000	Default	Early stopping	86.11	86.05	84.96	5 min 19s
Model 1D	LSTM	2 Layer LSTM _Unidirectional 64 + 32 Cells + 1 Dense Layer 64 Nodes	1000	Default	Early stopping + Dropout	84.77	86.35	85.03	6 min 3 s
Model 1AA	LSTM	1 Layer LSTM _Bidirectional 64 Cells + 1 Dense Layer 64 Nodes	2000	Default	Early stopping	88.59	88.07	87.69	18 min 42 s
Model 1BB	LSTM	2 Layer LSTM _Bidirectional 64 + 32 Cells + 1 Dense Layer 64 Nodes	2000	Default	Early stopping + Dropout	88.54	88.85	87.29	23 min 29 s
Model 1CC	LSTM	1 Layer LSTM _Unidirectional 64 Cells + 1 Dense Layer 64 Nodes	2000	Default	Early stopping	85.31	85.47	84.64	6 min 12 s
Model 1DD	LSTM	2 Layer LSTM _Unidirectional 64 + 32 Cells + 1 Dense Layer 64 Nodes	2000	Default	Early stopping + Dropout	83.3	88.96	87.43	10 min 23 s
Model 1AAA	LSTM	1 Layer LSTM _Bidirectional 64 Cells + 1 Dense Layer 64 Nodes	3000	Default	Early stopping	90.02	88.9	88.64	4 min 27 s
Model 1BBB	LSTM	2 Layer LSTM _Bidirectional 64 + 32 Cells + 1 Dense Layer 64 Nodes	3000	Default	Early stopping + Dropout	89.03	89.12	88.71	4 min 0 s
Model 1CCC	LSTM	1 Layer LSTM _Unidirectional 64 Cells + 1 Dense Layer 64 Nodes	3000	Default	Early stopping	90.19	88.97	88.44	3 min 24 s
Model 1DDD	LSTM	2 Layer LSTM _Unidirectional 64 + 32 Cells + 1 Dense Layer 64 Nodes	3000	Default	Early stopping + Dropout	83.62	89.48	88.05	2 min 12s
Model 2A	RNN	1 Layer RNN _Bidirectional 64 Cells + 1 Dense Layer 64 Nodes	1000	Default	Early stopping	86.26	86.3	85.22	9 min 55 s
Model 2B	RNN	2 Layer RNN _Bidirectional 64 + 32 Cells + 1 Dense Layer 64 Nodes	1000	Default	Early stopping + Dropout	84.29	86.15	83.61	7 min 7 s
Model 2C	RNN	1 Layer RNN _Unidirectional 64 Cells + 1 Dense Layer 64 Nodes	1000	Default	Early stopping	86.38	85.92	84.4	9 min 17 s
Model 2D	RNN	2 Layer RNN _Unidirectional 64 + 32 Cells + 1 Dense Layer 64 Nodes	1000	Default	Early stopping + Dropout	81.5	84.01	81.74	10 min 2 s
Model 2AA	RNN	1 Layer RNN _Bidirectional 64 Cells + 1 Dense Layer 64 Nodes	2000	Default	Early stopping	87.9	89.16	87.54	42 min 32 s
Model 2BB	RNN	2 Layer RNN _Bidirectional 64 + 32 Cells + 1 Dense Layer 64 Nodes	2000	Default	Early stopping + Dropout	87.17	86.98	85.92	37 min 17 s
Model 2CC	RNN	1 Layer RNN _Unidirectional 64 Cells + 1 Dense Layer 64 Nodes	2000	Default	Early stopping	88.92	87.8	87.17	26 min 43 s
Model 2DD	RNN	2 Layer RNN _Unidirectional 64 + 32 Cells + 1 Dense Layer 64 Nodes	2000	Default	Early stopping + Dropout	86.13	87.5	85.2	22 min 5 s
Model 2AAA	RNN	1 Layer RNN _Bidirectional 64 Cells + 1 Dense Layer 64 Nodes	3000	Default	Early stopping	90.81	88.75	88.39	36 min 17 s
Model 2BBB	RNN	2 Layer RNN _Bidirectional 64 + 32 Cells + 1 Dense Layer 64 Nodes	3000	Default	Early stopping + Dropout	90.08	88.02	86.78	25 min 42 s
Model 2CCC	RNN	1 Layer RNN _Unidirectional 64 Cells + 1 Dense Layer 64 Nodes	3000	Default	Early stopping	90.59	88.72	87.91	16 min 28 s
Model 2DD	RNN	2 Layer RNN _Unidirectional 64 + 32 Cells + 1 Dense Layer 64 Nodes	3000	Default	Early stopping + Dropout	79.21	86.98	86.96	7 min 2 s
Model 3A	CNN	1 layer CON1d 64 Kernel, MaxPool1D, 3, 1 Dense layer 64 Nodes	3000	40	Early stopping	89.91	89.15	88.83	4 min 42 s
Model 3B	CNN	3 Layer CON1d 64 + 128 + 256 Kernel , MaxPool1D, 2 Dense layer 256 + 256 Nodes	3000	40	Early stopping + Dropout	90.43	88.75	87.81	16 min 12s
Model 3C	CNN	1 layer CON1d 64 Kernel, MaxPool1D, 3, 1 Dense layer 64 Nodes	Max	60	Early stopping + DropoutEarly stopping + Dropout	94.25	91.08	90.68	1 min 37s
Model 4A	LSTM	1 Layer LSTM _Bidirectional 128 Cells + 1 Dense Layer 128 Nodes	1000	60	Early stopping + Dropout	87.38	87	88.36	5 min 4 s
Model 4B	LSTM	1 Layer LSTM _Bidirectional 128 Cells + 1 Dense Layer 128 Nodes	3000	60	Early stopping + Dropout	86.07	89.38	88.86	12 min 38s
Model 4C	LSTM	1 Layer LSTM _Bidirectional 64 + 32 Cells + 1 Dense Layer 128 Nodes	3000	40	Early stopping + Dropout	92.46	90.42	88.52	11 min 50 s
Model 4D	LSTM	1 Layer LSTM _Bidirectional 128 Cells + 1 Dense Layer 128 Nodes	Max	60	Early stopping + Dropout	8.29	92.6	89.63	3 min 2 S
Model 5A	RNN	1 Layer RNN _Bidirectional 64 Cells + 1 Dense Layer 64 Nodes	3000	40	Early stopping + Dropout	90.26	88.78	87.1	8 min 25 s
Model 5B	RNN	2 Layer RNN _Bidirectional 64 + 32 Cells + 1 Dense Layer 64 Nodes	3000	40	None	81.71	88.23	87.6	21 min 8 s

Appendix B

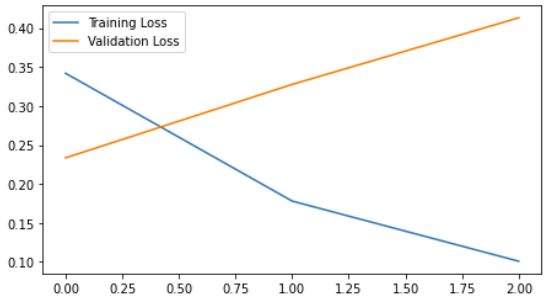
BEST model 3C

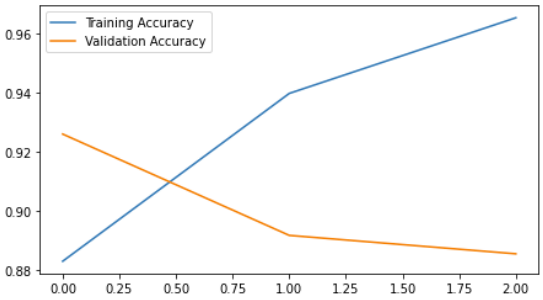




Appendix C

LSTM best Model 4D






```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
text_vectorization (TextVect (None, 60)                0
_________________________________________________________________
embedding (Embedding)        (None, 60, 128)           11622400
_________________________________________________________________
bidirectional (Bidirectional (None, 256)               263168
_________________________________________________________________
dropout (Dropout)            (None, 256)               0
_________________________________________________________________
dense (Dense)                (None, 128)               32896
_________________________________________________________________
dropout_1 (Dropout)          (None, 128)               0
_________________________________________________________________
dense_1 (Dense)              (None, 4)                 516
=================================================================
Total params: 11,918,980
Trainable params: 11,918,980
Non-trainable params: 0
_________________________________________________________________
```