Big Data Summative Assignment

hfsk44

March 11, 2018

Text Cleaning

A variety of methods were used to convert the text corpus to a standard form, such that it could be more easily modelled by a classifier. Duplicate texts were first removed, as that would otherwise lead to the model overfitting on those texts. Texts of less than 10 characters in length were also removed as they would not contain enough information for any meaningful inference to be made about their validity.

Since binary classification is a matter of understanding text on a macroscale, as opposed to a microscale task such as information extraction, the text cleaning process can be reasonably harsh. As such, the pipeline consisted of removing HTML tags, converting the text to ASCII and then removing all non-alphanumeric characters besides spaces and mid-word hyphens.

While the corpus contained a large number of hyperlinks and twitter handles, they were not explicitly filtered out as a model could reasonably learn a certain user or website to be untrustworthy. Words were also initially lemmatized, but in preliminary testing with a naive Bayes classifier, this was found to reduce classification accuracy and was therefore removed from the pipeline.

Below is an example of the text cleaning process on a single corpus item. It can be seen that the resulting text is in a significantly more standardised format, while still giving enough information for a macroscale understanding.

Original:

`With little fanfare this fall, the New York developer who had planned to build an Islamic community center north of the World Trade Center announced that he would instead use the site for a 70-story tower of luxury condos. Those who had rallied in opposition to the building because of its religious affiliation back in 2010 were exultant. The importance of the defeat of the Ground Zero Mosque cannot be overstated, Pamela Geller, president of the American Freedom Defense Initiative, wrote on the website Breitbart in September. The Ground Zero Mosque became a watershed issue in our effort to raise awareness of and ultimately halt and roll back the advance of Islamic law and Islamic supremacism in America.`

Cleaned:

`little fanfare fall new york developer planned build islamic community center north world trade center announced would instead use site 70-story tower luxury condos rallied opposition building religious affiliation back 2010 exultant importance defeat ground zero mosque cannot overstated pamela geller president american freedom defense initiative wrote website breitbart september ground zero mosque became watershed issue effort raise awareness ultimately halt roll back advance islamic law islamic supremacism america`

Shallow Learning

The corpus was split into 4843, 968, and 1213 samples for training, validation and testing respectively. This corresponds to an 80/20 split for training and testing, with 20% of the training set held out for validation. This allows for the models to train on as much data as possible, while ensuring that there are enough samples in the validation and testing set for the results to be statistically reliable.

Feature extraction and classification were performed with the TfidfVectorizer and MultinomialNB classes from scikit-learn. The benefits of using a reputable library over a homemade implementation are numerous. Beyond saving development time, the implementations have been thoroughly tested, optimised, and documented, ensuring that the documents are classified as they should be in a timely manner.

A grid search was performed on the maximum n-gram size and document frequency cut-offs of the TfidfVectorizer, as well as whether IDF values were included. Each vectorizer configuration was evaluated on the validation results of a naive Bayes classifier trained on the generated feature vectors. The F1 measure was chosen as the primary evaluation method for a model as the task places no preference over precision or recall. It is important that we filter out only fake news, but equally important that all of the fake news is identified.

Results

As shown in Table 1, of the 137 configurations tested, feature vectors formed from term frequency scores resulted in higher F1 values than TF-IDF. Allowing n-grams of a size larger than one proved to be beneficial but a maximum n-gram size greater than two yielded no noticeable increase in performance. This is due to the vectorizer determining larger n-grams to be insignificant, regardless of if it is permitted to include them. With bi-grams permitted, the best performing vectorizer's vocabulary contained only nine bigrams and 534 unigrams.

Ignoring terms with a normalised document frequency lower than 0.1 proved to be highly beneficial, with the top 41 configurations using this cut-off. This is effective in reducing the size of the vocabulary from 1,229,620 to 543 terms. Intuitively, using a set of terms that appear in a large number of texts will prevent overfitting as the model is forced to learn the general case, instead of learning to classify any article with the n-gram 'trump did nothing wrong' as fake. Conversely, higher cut-offs are likely to underfit as they will consist of corpus specific stopwords that appear so often that they provide no real information. A maximum document frequency threshold was also tested but this did not have a noticeable effect on results.

The best performing shallow classifier identified in testing used term frequencies on 1-2 grams, with a document frequency cut-off of 0.1. When run on the testing set, this yielded an F1 score of 85% with precision and recall scores of 83% and 86% respectively, a very reasonable result for such a simple classification method. Figure 2 shows a visualisation of the false positives and false negatives for this classifier on the testing set.

Method	Max n-gram size	Min DF	Max DF	Precision	Recall	F1
tf	2	0.1	0.75	0.858	0.866	0.862
tf	2	0.1	0.9	0.858	0.866	0.862
tf	2	0.1	1	0.858	0.866	0.862
tf	3	0.1	0.75	0.858	0.866	0.862
tf	3	0.1	0.9	0.858	0.866	0.862
tf	3	0.1	1	0.858	0.866	0.862
tf	5	0.1	0.75	0.858	0.866	0.862
tf	5	0.1	0.9	0.858	0.866	0.862
tf	5	0.1	1	0.858	0.866	0.862
tf	1	0.1	0.75	0.853	0.864	0.859
		⋮				
tf	2	0	0.9	0.992	0.495	0.661
tf	2	0	1	0.992	0.495	0.661
tf	3	0	0.5	0.996	0.475	0.644
tf	3	0	0.75	0.996	0.454	0.623
tf	3	0	0.9	0.996	0.454	0.623
tf	3	0	1	0.996	0.454	0.623
tf	5	0	0.5	0.996	0.446	0.616
tf	5	0	0.75	1.000	0.424	0.596
tf	5	0	0.9	1.000	0.424	0.596
tf	5	0	1	1.000	0.424	0.596

Table 1: Results of a grid search for configuring the TfidfVectorizer. Parameters corresponding to the 10 highest and lowest F1 scores are shown.

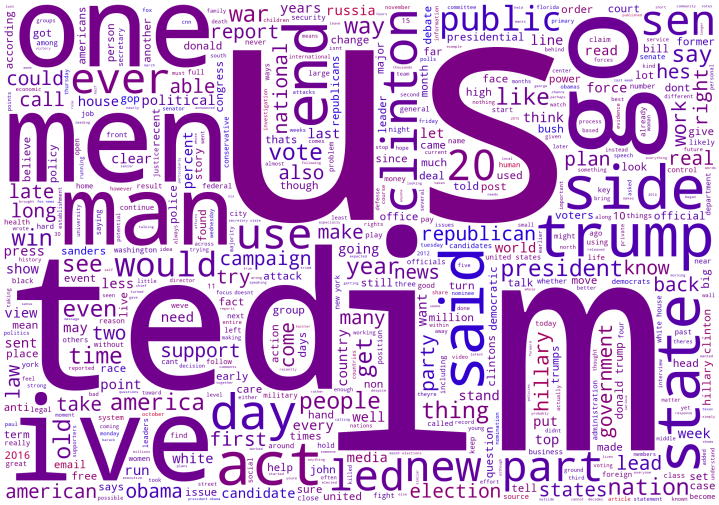
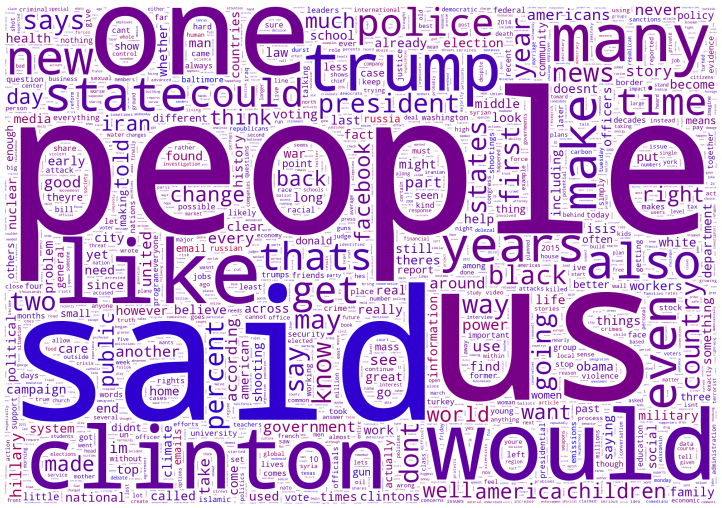


Figure 1: Vocabulary wordcloud for the best performing TF-IDF vectorizer, using term frequency of 1-2 grams and minimum DF of 0.1. Size of a term is relative to document frequency, color is relative to whether the word is present in predominantly fake (red) or real (blue) texts.



(a)

(b)

Figure 2: Wordclouds for false positives (a) and false negatives (b) for the best performing shallow classifier. Notable mentions include 'black', 'iran', and 'police' in (a) and 'white', 'obama', and 'republican' in (b).

Deep Learning

Word2Vec

For the deep learning classifiers, feature vectors were created with the pre-trained Google News Word2Vec model. A single input text is split into its component words, which are then replaced with the word's index in the Word2Vec model's vocabulary. This format allows for the use of a Keras embedding layer, which can, if the correct embedding matrix is supplied, efficiently convert the vocabulary indices into their corresponding vector representation.

The Google News model was chosen because of the nature of the data it was trained on. Compared to models trained on other sources, such as the Twitter or the Wikipedia corpora, the Google News model is more likely to effectively represent political relationships between words, since a high proportion of news

articles are political in nature. These political relationships would be highly valuable for differentiating fake news from real, as the issue of fake news is political in its nature. For the sake of saving memory, the actual model used was trained on the subset of articles written in English. As the vast majority of the fake news corpus is English, this is a reasonable trade-off.

LSTM

Since the majority of problems can be modelled effectively with a single hidden layer (Heaton 2008, p158), the general structure of the LSTM model is simple, as shown in Figure 3. Through the embedding layer, a single corpus item is converted to a 1000 long sequence of 300 wide feature vectors, padded with a masking value of zero if necessary. This is then passed to a single LSTM layer, using tanh as the activation function, and further on to the output layer which is a single neuron with a sigmoid activation function.

The choice of tanh over sigmoid as the LSTM layer activation is due to the fact that sigmoid is non-decreasing, whereas tanh lies between -1 and 1. This means that on average, the activation will be zero for a layer using tanh, and positive for sigmoid, meaning that tanh should be expected to converge at a faster rate (LeCun et al. 1998). For the output neuron, sigmoid is the obvious choice of activation as it can map directly to a probability between 0 and 1.

For selecting the most effective hyperparameters to the model, a number of grid searches were performed, each exploring a new set of parameters on the best performing configurations from the last search. The initial search tested variations on batch size, the number of neurons in the LSTM layer, and the learning rate of the adam optimiser (Table 2). Results looked promising, with the best configuration giving an F1 score of 94%. In general, it was found that a learning rate of 0.005, batch size of 32 and small layer size were the most effective. A notable result is that the highest F1 score belongs to an LSTM layer with only eight neurons, appearing to defy the standard rule-of-thumb of using a layer size of roughly 2/3 the input size. This suggests that the classification problem is close to linearly separable, an idea that is further supported by the relatively strong performance of the naive Bayes classifier, which is equivalent to a linear classifier in log space (Rennie et al. 2003).

The five best performing configurations were selected for further testing, varying the dropout rate on the LSTM layer. This did not yield any particularly interesting results, with the default of 20% giving the two highest F1 scores (Table 3). From here, the optimisation method was varied on the two best performing configurations from round 2 (Table 4), showing that the default adam optimizer was by far the best solution.

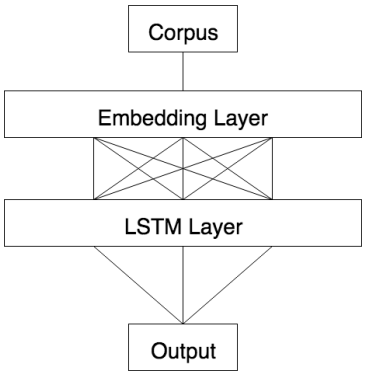


Figure 3: Diagram of the basic LSTM model

Learning rate	Batch size	Layer size	Precision	Recall	F1
0.005	32	8	0.952	0.929	0.940
0.005	32	16	0.932	0.947	0.940
0.005	32	32	0.947	0.910	0.928
0.005	128	64	0.932	0.912	0.922
0.005	32	256	0.928	0.910	0.919
0.005	32	64	0.919	0.912	0.915
0.005	32	128	0.892	0.912	0.902
0.005	128	256	0.902	0.902	0.902
0.005	128	128	0.920	0.882	0.901
0.0005	512	128	0.869	0.833	0.851
		⋮			
0.0005	32	256	0.813	0.699	0.752
0.005	512	256	0.847	0.654	0.738
0.005	512	32	0.592	0.976	0.737
0.0005	32	16	0.584	0.980	0.732
0.0005	2048	32	0.920	0.607	0.731
0.0005	2048	8	0.912	0.609	0.730
0.0005	128	256	0.787	0.623	0.695
0.0005	32	32	0.928	0.454	0.609
0.001	128	128	0.876	0.428	0.575
0.0005	128	128	0.922	0.369	0.527

Table 2: Results of the first grid search for configuring the LSTM model. Parameters corresponding to the 10 highest and lowest F1 scores are shown.

Dropout	Learning rate	Batch size	Layer Size	Precision	Recall	F1
0.2	0.005	32	8	0.952	0.929	0.940
0.2	0.005	32	16	0.932	0.947	0.940
0.6	0.005	32	32	0.967	0.910	0.937
0.4	0.005	128	64	0.942	0.925	0.934
0.4	0.005	32	8	0.921	0.943	0.932
0.6	0.005	32	16	0.949	0.916	0.932
0.4	0.005	32	32	0.951	0.910	0.930
0.2	0.005	32	32	0.947	0.910	0.928
0.4	0.005	32	256	0.919	0.931	0.925
0.4	0.005	32	16	0.932	0.912	0.922
0.2	0.005	128	64	0.932	0.912	0.922
0.2	0.005	32	256	0.928	0.910	0.919
0	0.005	32	32	0.906	0.929	0.918
0	0.005	32	256	0.900	0.916	0.907
0.6	0.005	32	256	0.945	0.872	0.907
0	0.005	32	16	0.944	0.866	0.904
0.6	0.005	32	8	0.928	0.841	0.882
0	0.005	128	64	0.879	0.829	0.853
0	0.005	32	8	0.859	0.778	0.816
0.6	0.005	128	64	0.782	0.811	0.797

Table 3: Results of the second grid search for configuring the dropout rate on the LSTM layer.

Learning rate	Batch size	Layer Size	Dropout	Optimizer	Precision	Recall	F1
0.005	32	8	0.2	adam	0.952	0.929	0.940
0.005	32	16	0.2	adam	0.932	0.947	0.940
0.005	32	16	0.2	rmsprop	0.950	0.904	0.926
0.005	32	8	0.2	rmsprop	0.905	0.882	0.894
0.005	32	16	0.2	adagrad	0.832	0.825	0.828
0.005	32	8	0.2	adagrad	0.789	0.868	0.827
0.005	32	8	0.2	sgd	0.741	0.692	0.715
0.005	32	16	0.2	sgd	0.879	0.100	0.180

Table 4: Results of the third grid search for configuring the optimizer.

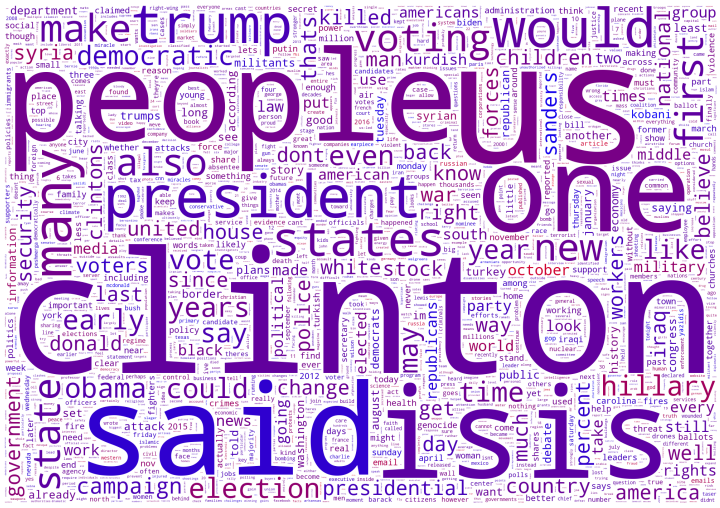
LSTM vs. RNN

The best performing LSTM classifier identified in testing used a batch size of 32, with a single LSTM layer of size 8, dropout of 20%, and an adam optimizer with a learning rate of 0.005. An equivalent RNN was created by replacing the LSTM layer of this model with a Keras SimpleRNN layer and training on the same dataset. Figures 5 and 6 show the loss and accuracies at each training epoch for the LSTM and RNN. To prevent overfitting, training was pre-empted after a plateau in validation loss greater than five epochs, the models were also checkpointed such that weights would be saved at the epoch with the highest validation accuracy.

When run on the testing set, the LSTM yielded an F1 score of 90% with precision and recall scores of 90% and 89% respectively. Figure 4 shows a visualisation of the false positives and false negatives for this classifier on the testing set. The RNN, on the other hand, yielded an F1 score of 73% with precision and recall of 71% and 76%. While it is likely the case that further parameter tuning on the RNN would yield better results, it would appear that it is vastly outperformed by the LSTM. A possible explanation for this is that the RNN's performance is hampered by the 'vanishing gradient problem', whereas by design, the LSTM is not (Pascanu, Mikolov, and Bengio 2013). Since fake news classification is a macroscale task, it seems intuitive that a classifier with a lessened ability to learn correlations between temporally distant elements would indeed struggle to perform well. The only redeeming feature for the RNN is the shorter training time of 53s compared to 208s per epoch, but this is not enough to justify its use over the LSTM.

Conclusion

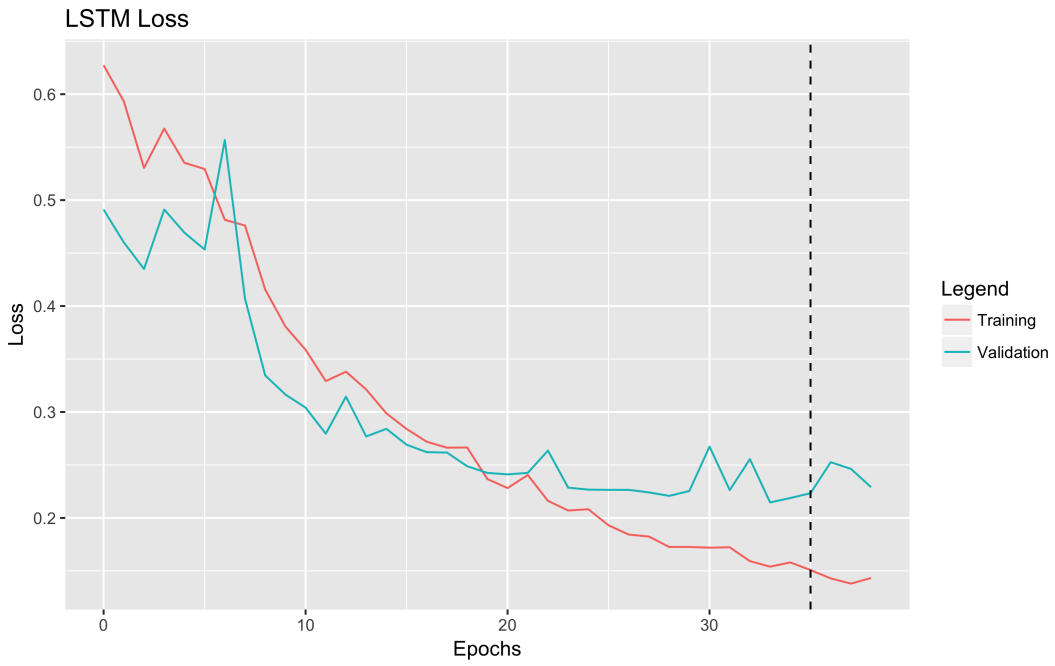
While a deep learning approach did eventually outperform the shallow classifier, it must be noted that the naive Bayes method gave acceptable results almost immediately, whereas the LSTM required an arduous parameter tuning process. This is understandable as the problem is highly similar to that of spam detection, the quintessential application of a naive Bayes classifier. However, with some fine tuning, a deep learning approach managed to beat Bayes on its home turf. This is a testament to the power of deep learning, and its ability to effectively solve a wide range of problems, often to a higher level of success than purpose-built algorithms.



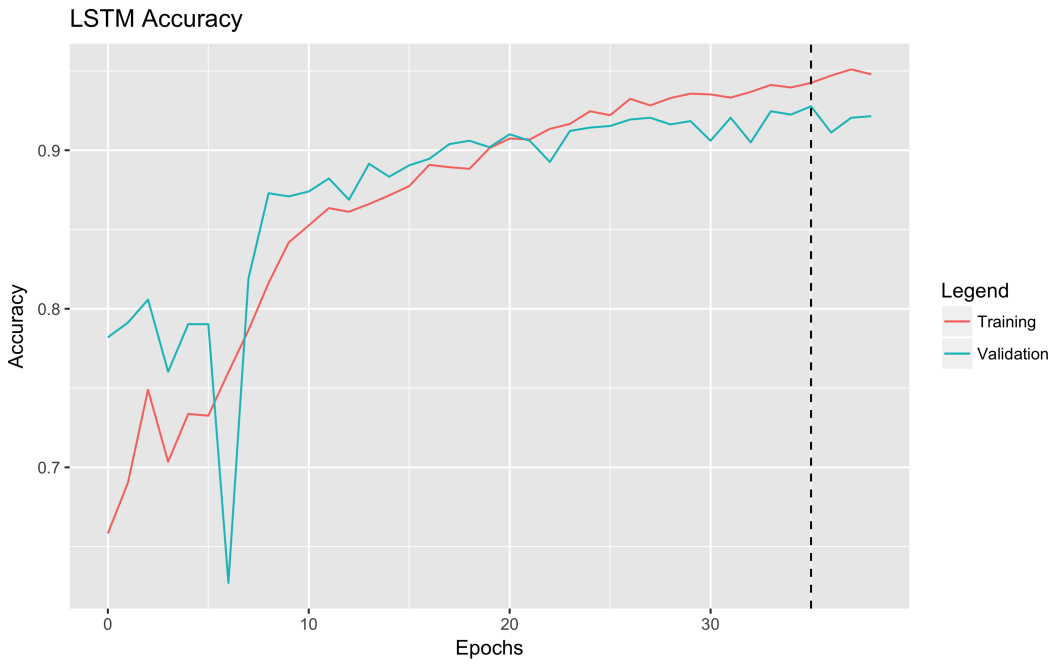
(a)

(b)

Figure 4: Wordclouds for false positives (a) and false negatives (b) for the best performing LSTM classifier. Notable mentions include 'yglesias' - the surname of a U.S politics journalist, and 'splc' - the initials of the Southern Poverty Law Center, a U.S based non-profit organisation that monitors hate groups in (b).

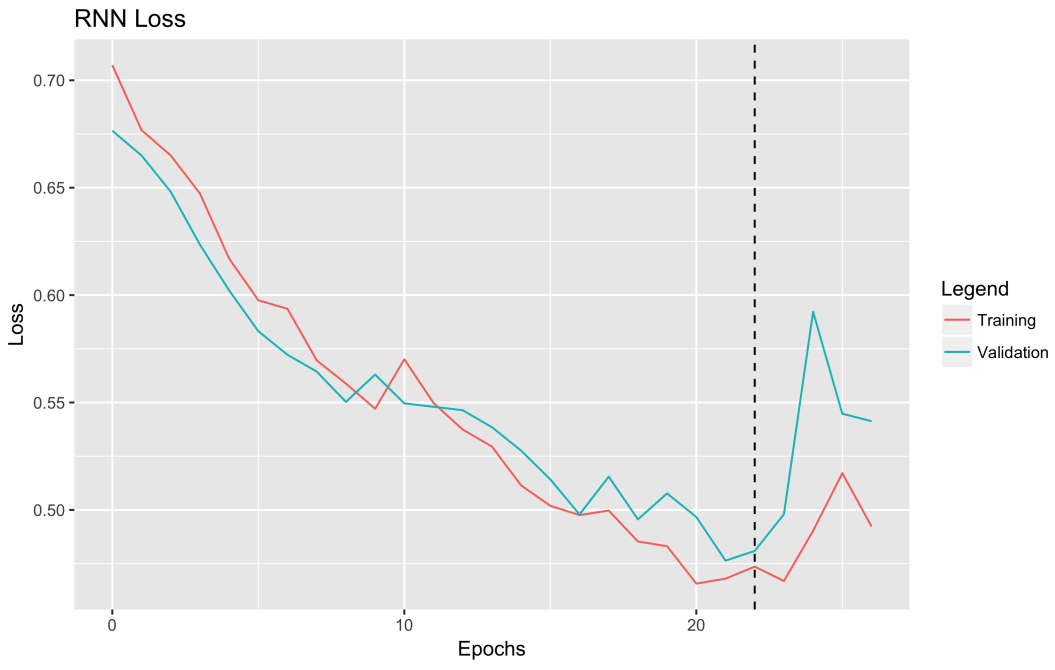


(a)

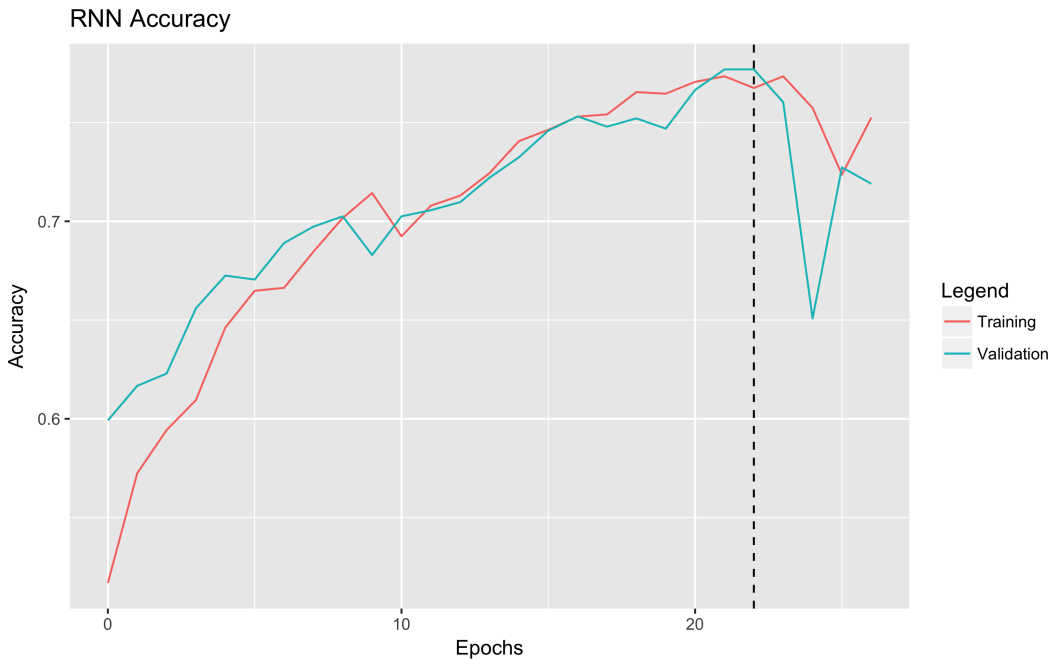


(b)

Figure 5: Loss and accuracy plots for training the final LSTM model. Dashed line indicates where the model was saved



(a)



(b)

Figure 6: Loss and accuracy plots for training the final RNN model. Dashed line indicates where the model was saved

References

Heaton, Jeff (2008). *Introduction to neural networks with Java*. Heaton Research, Inc.

LeCun, Yann et al. (1998). "Efficient backprop". In: *Neural networks: Tricks of the trade*. Springer, pp. 9–50.

Pascanu, Razvan, Tomas Mikolov, and Yoshua Bengio (2013). "On the difficulty of training recurrent neural networks". In: *International Conference on Machine Learning*, pp. 1310–1318.

Rennie, Jason D et al. (2003). "Tackling the poor assumptions of naive bayes text classifiers". In: *Proceedings of the 20th international conference on machine learning (ICML-03)*, pp. 616–623.