Machine Learning Engineer Nanodegree

Final project report: Classification of news as reliable versus unreliable

Amin Momin

June 10, 2018

Project Overview

Natural language processing (NLP) is a growing field that aims to analyze and understand textual information. It has been successfully applied for sentiment analysis, spam email detection, text classification and language translation (1,2,3,4,5). The common approaches to perform NLP is using Bayesian statistical models such as naive bayes and deep learning frameworks such as recursive neural networks (RNN). In the current project we applied NLP algorithm to classify unreliable news stories.

Problem Statement

Fake news or unreliable news is often published for political or monitory gains by unreliable or nefarious sources (6). The proliferation of blogs, social media, and image and video sharing has exponentially increased sources of fake information. Unlike professional news organizations, these crowd sourcing and anonymous platforms disseminate information with few checks for authenticity. This has impacted hate speech, public policy, elections and safety.

Facebook and Google have been the most prominent examples of online platform often used to spread false article and headline. During the 2016 US presidential elections foreign sources spread millions of fake news articles targeting 126 million Facebook users in the United States (7). A study by Buzzfeed discovered that the fake news articles received more views than true news stories (8)

To counter this false propaganda assault on democracy the current project aims to build an algorithm to classify news articles and statements as authentic versus fake using machine learning and deep learning approaches. These methods will have the potential to understand the sentiment and context of statement and classify the category of the news articles.

Dataset

For the present project I will use a dataset provided by a Kaggle project Fake news classification challenge (https://www.kaggle.com/c/fake-news). The training subset contain 20800 text articles and labels, while the test data contains 5200 similar information without the label. The goal of the project is to build a suitable classification model using the training data and validates the findings on the test dataset. The dataset has been manually curated with the labels. Since the test dataset doesn't have label 20% of the training articles were retained for testing.

- id: unique id for a news article
- title: the title of a news article
- author: author of the news article
- text: the text of the article; could be incomplete
- label: a label that marks the article as potentially unreliable
    - 1: unreliable
    - 0: reliable

The dataset contains relevant input information to classify the articles, such as title of the article, authors name and text of the article. Unreliable article frequently have titles with controversial text or attention grabbing keywords to get the reader's attention. Certain authors are frequently associated with unreliable sources. In addition, poorly written article have frequently associated with un-reputed news sources. Thus, having the variable of title, author and article text are valuable inputs for building news classification model using machine learning approaches.

Metrics

Once the predictions for the test article are generated using the RNN-DL model, they will be compared to the actual labels. The evaluation will be based on the total correctly classified articles compared to the total number of articles using precision (P), recall (R) and F beta-score (9,10, 11).

$$P = \frac{TP}{TP + FP}$$

$$R = \frac{TP}{TP + FN}$$

$$f\ beta - score\ =\ (1\ +\ \beta ** 2) * \frac{precision\ *\ recall}{(\beta ** 2) * precision\ +\ recall}$$

Precision measures the predicted true positives compared to all the predicted positives. This is critical in applications where misclassification of a positive can have a consequence. Example, spam email classification of a legitimate message can result in loss of important information.

Recall on the other hand compared predicted true positives to all positives. This statistic is crucial in situations where misclassification of a positive as a false negative can be detrimental. For example, misclassification of sick patient as healthy can have very serious health consequences.

F beta-score provides a measure between precision and recall. It is the weighted harmonic mean of precision and recall. Beta < 1 gives greater wieight to precision , while beta > 1 gives higher weight to recall.

II. Analysis

Data Exploration

For the current project we plan to use the training dataset containing 20800 articles for both training and testing. These articles were manually labeled as reliable or unreliable (labeled 0 and 1 respectively). Prior to performing NLP analysis, the data was explored to understand broad trend using the python Pandas library.

In the initial cleaning step, 370 articles with empty strings in the dataset were discarded. Subsequently, the number of article and authors for each label were evaluated (Fig 1). For reliable articles (label 0) the number of authors and articles were about the same. In contrast the unreliable (label 1) articles had fewer authors.

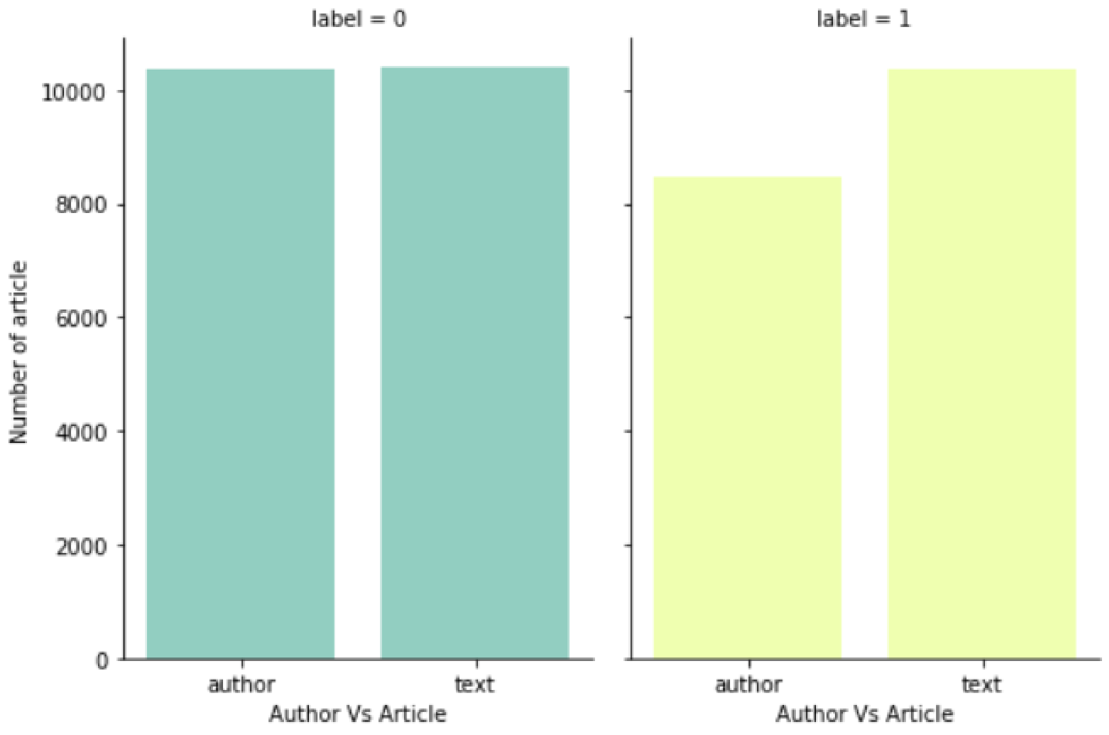


Fig1: Comparison of articles and authors for each type of article (reliable ‘0’, unreliable ‘1’)

In addition, the language of the remaining articles was analyzed using the langdetect library. The analysis (Fig 2) detected that the dataset contains predominantly English (en) article for both reliable and unreliable sources. This is critical since the embedding of text vector for NLP is highly language specific.

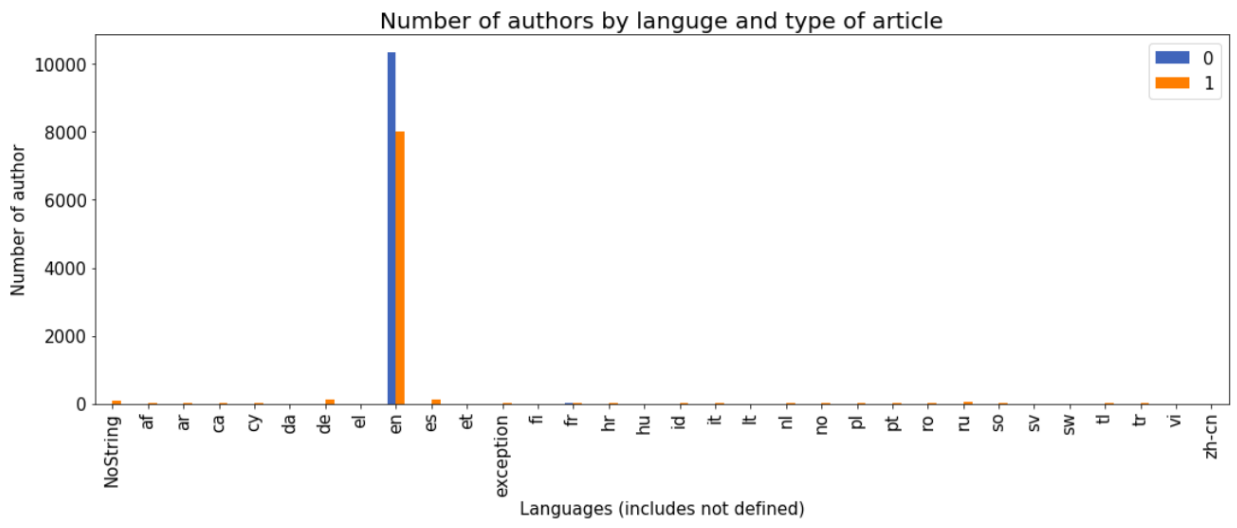


Fig 2: Analysis of language of the article within the Kaggle fake news dataset

Algorithms and Techniques

In the current project we will apply recurrent neural network (RNN) to analyze text data for classification. RNN are neural network that analyze sequence of word or speeches and help understand the content and context of the language (12). Each word of a sentence is analyzed sequentially in the input order by a neuron that applies an activation. The output is passed on to the next neuron that analyzes the next word and calculates the output based on the current word and input from all previous words (Fig 3) .

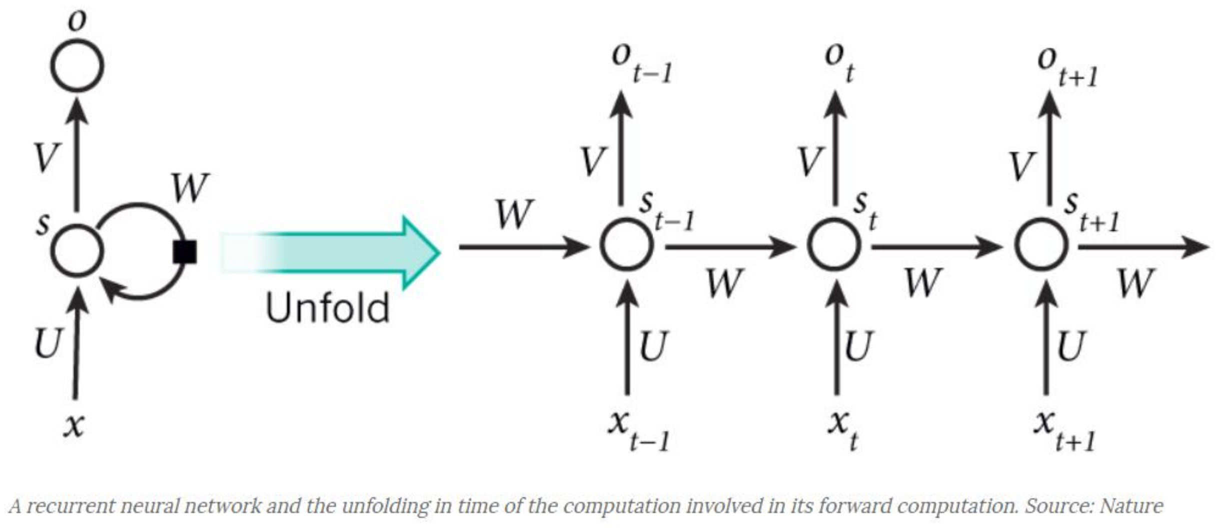

o
V
s
W
U
x
Unfold
W
o t−1
V
s t−1
U
x t−1
W
o t
V
s t
U
x t
W
o t+1
V
s t+1
U
x t+1
W
A recurrent neural network and the unfolding in time of the computation involved in its forward computation. Source: Nature

Fig 3: Depiction of recurrent neural network (RNN) of a sequence of states. The figure is acquired from a blog, wildml (12).

Long short term memory (LSTM) is a form of RNN that remembers short and long term connections between words of the statement. This is achieved by building cells that can selectively remember of forget prior input words in the text corpus (Fig 4). It has resulted in significant improvements in analysis of language, sentiment, classification and language translation.

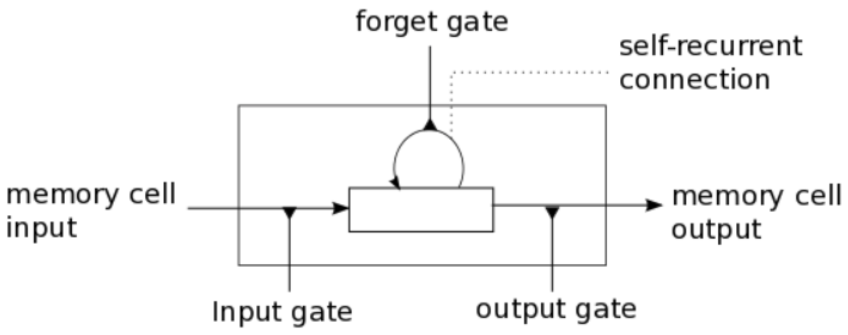


Fig 5: Depiction of a LSTM memory cell with input and output gate. Figure taken from (13)

Bidirectional LSTM: While LSTM models read the words of a text from start to end, they may miss important relationship that may be provided by words on the opposite side of the sentence. To address the issue of learning relationships within word sentences at either end of the statement the bidirectional method was invented. This method tries to learn the relationships on either end of the text sentence. It is implemented as a single method in Keras.

Word tokenization: Prior to using RNN models the word in text are vectorized as numbers using one hot encoding. Where every word in the corpus is given a unique integer. This is essential since matrix multiplication can only be done on numbers. Usually for small studies 20000 to 40000 words are used. On the other hand larger commercial language translation systems use a million words. However, as the vector length increases the computation gets extremely expensive given the sparse nature of the vector.

Word embedding: Word embedding was discovered to overcome the issues with sparse long one-hot token vectors. It identified word similarity based on co-occurrence of words within a window. The process of word embedding is carried out on a text corpus of millions to a billion words for learning the word embeddings. Multiple different approaches for word embedding include word2vec, continuous bag of words and GloVe . Word embedding not only reduces the dimensionality from one hot encoding to smaller n-dimensional vector, but it is very effective in identify similar words. The word embedding is represented in n-dimensional vector space by multidimensional scaling and hierarchical clustering (Fig 6)

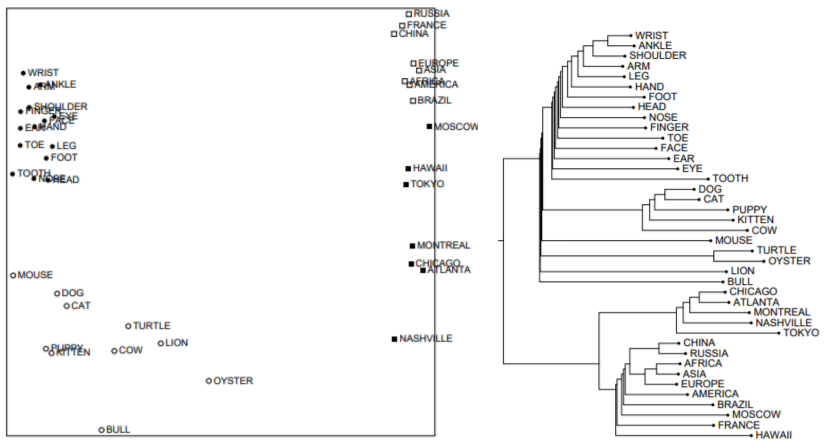


Fig 6: Multidimensional scaling and hierarchical clustering view of Word embedding representation (14)

Benchmark

In the current study we build a model a model with subset of the original data for testing the deeplearning pipeline using 2000 articles for training and 500 articles to test. This initial model had training accuracy of, and testing accuracy of 0.8650 and testing accuracy of 0.85. The model was trained for 10 epochs. The model testing had a recall (0.8714), precision (0.8346) and F-beta score (0.8417).

III. Methodology

Data Preprocessing

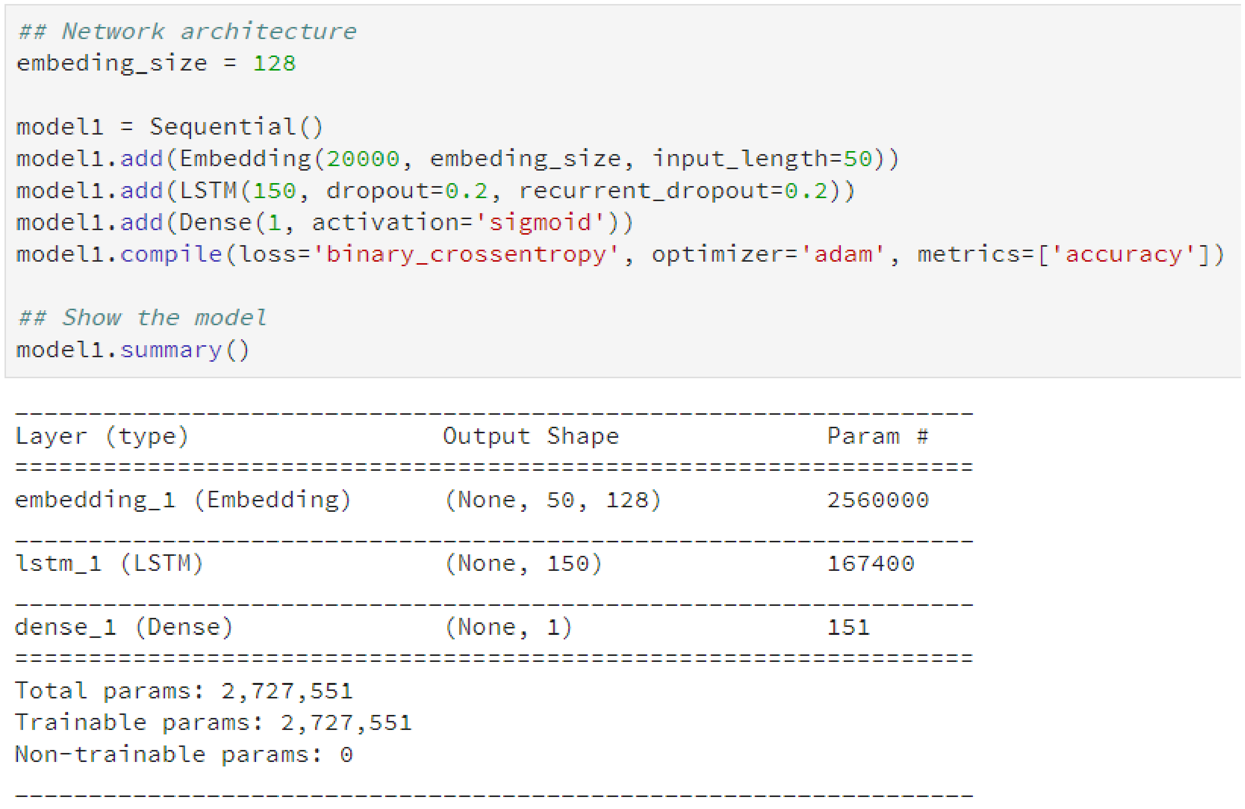
Data preprocessing is the most critical step of NLP analysis pipeline. It requires great accuracy and insight of the data to prepare the data. This is specifically challenging for textual information since the string of words must be transformed to numeric vectors for training using neural nets or other statistical methodologies. The preprocessing involves data cleaning, tokenization and word embedding. In addition, for the current project we performed language detection in order to eliminate non English articles.

Removal of non-English articles: Following the analysis of language of the article rows corresponding to non-English articles were dropped. Additionally, articles with empty strings were removed from the data set. This reduced the size of the entire dataset to 20099 articles.

Data cleaning: In this step the dataset articles were scanned for punctuations, stop words, which were removed. The strings were then all converted to lower case. The clean text was tokenized to a vocabulary size of 20000. All these steps were performed using a keras function ‘tokenize’. The tokenization converts the string to token vectors of length 20000. The articles were subsequently split into train and testing set. The training and validation set includes the first 15000 articles. The remaining 5099 articles were used for the testing the model.

Implementation

The RNN/LSTM model for the text classification was implemented in Keras using the tensorflow backend. The sequential model framework was used to build the neural net architecture. The neural network architecture includes word embedding, LSTM and the dense output layers (Fig 7).



```
## Network architecture
embeding_size = 128

model1 = Sequential()
model1.add(Embedding(20000, embeding_size, input_length=50))
model1.add(LSTM(150, dropout=0.2, recurrent_dropout=0.2))
model1.add(Dense(1, activation='sigmoid'))
model1.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])

## Show the model
model1.summary()
```

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
embedding_1 (Embedding)      (None, 50, 128)           2560000   
_________________________________________________________________
lstm_1 (LSTM)                (None, 150)               167400    
_________________________________________________________________
dense_1 (Dense)              (None, 1)                 151       
=================================================================
Total params: 2,727,551
Trainable params: 2,727,551
Non-trainable params: 0
_________________________________________________________________
```

Fig 7: Depiction of the RNN/LSTM neural net implemented for the current project using Keras with tensorflow backend.

Word embedding: The first step in the model building is to construct word embedding vectors. For the current project the dataset text were embedded into 128 dimension vector. This process of embedding words as dense vector reduces the dimensionality and makes the computation efficient (15). Using higher dimension embedding vector can make the deeplearning process more accurate however it increases the computation cost and thus slows the model training.

Additionally, only the first 50 words of the article were used for the model construction. Padding of the sequences was performed for any article less than 50 words in order to make all the training vector of the same sequence length.

LSTM layer: The LSTM layers was applied with or without bidirectional method was applied to the word embedding. This layer is the critical to building the deeplearning model with memory cells to understand the articles sentences as well as their context. The LSTM layers used has 100/150 memory cells. Additionally, It has a dropout of 0.2.

The final layer of the model is a single dense layer with a sigmoid activation. This activation is consistent with the problem to obtain a binary classification of the article as reliable or unreliable (0 and 1 respectively).

The Keras model was compiled using bionary_crossentropy as the loss function and 'adam' optimizer and used the 'accuracy' metric.

- Loss function: The loss function determines the difference between the true and predicted label. For binary classification, binary_cross entropy is an appropriate option
- Optimizer: Adam is a popular optimizer for reducing the loss function
- Metric: Accuracy is an appropriate metrics for classification since it calculates the correct prediction compared to the total counts.

Refinement

During the refinement process of the model, different parameters were modified to improve the deeplearning model's accuracy as depicted in Table 1. This included the sample size, LSTM cells, bidirectional layer for LSTM etc.

Stage 1: The initial model was built and test with a subset of the data comprising 2000 article for training and 500 for testing. The model was built with an LSTM layer of 100 unit and trained for 10 epochs

. The model showed accuracy between 85 – 86%

Stage 2: The model was trained with 15000 articles and tested with 5500 articles. The model accuracy had a learning layer of LSTM or LSTM + bidirectional. All other parameters were kept constant. The model gained accuracy to 88 - 89 %.

Stage 3: Further increment to LSTM layer with 150 cells didn't increase the accuracy of the model performance.

Stage 4: Increment of the Epoch to 25 for the model didn't improve the model test accuracy.

IV. Results

LSTM RNN models

The model to classify new article as reliable vs unreliable was constructed using the LSTM cells for the deeplearning pipeline. The analysis was initially constructed with a subset of the data, then trained with the entire data as depicted in Table 1. The initial increase in the training data from 2000 to 15000 article improved the models accuracy (Train Ac) from 86% to 89%. Concurrently the F beta-score also improve from 85 % to 89 %. This was achieved while using a single LSTM layer.

The use of bidirectional LSTM as well as 150 memory cell Vs 100 didn't show significant improvements in the model accuracy. Additionally, the selection of English only articles didn't change the model accuracy.

No	**Model**	**Sample Train/Test**	**Lang**	**LSTM cells**	**Eph**	**Train Ac**	**Test Ac**	**Recall**	**Precision**	**F beta**
1	LSTM/RNN	2000/500	All	100 (1),	10	0.8650	0.85	0.8714	0.8346	0.8417
2	bLSTM/RNN	2000/500	All	100 (1),	10	0.8525	0.84	0.7550	0.9038	0.8695
3	LSTM/RNN	15000/5500	All	100 (1),	10	0.8870	0.47	0.5409	0.4707	0.4832
4	bLSTM/RNN	15000/5500	All	100 (1),	10	0.8890	0.88	0.8595	0.9053	0.8958
5	LSTM/RNN	15000/5500	All	150 (1),	10	0.8813	0.88	0.8861	0.8791	0.8805
6	bLSTM/RNN	15000/5500	All	150 (1),	10	0.8863	0.89	0.8752	0.8940	0.8902
7	LSTM/RNN	15000/5099	English	150 (1),	10	0.8897	0.89	0.8769	0.8920	0.8890
8	bLSTM/RNN	15000/5099	English	150 (1),	10	0.8907	0.89	0.8818	0.8825	0.8823
9	LSTM/RNN	15000/5099	English	150 (1),	25	0.8937	0.88	0.8727	0.8856	0.8830

Table 1: Summary of the results for different model to training and testing the LSTM network for news classification. Eph: Epoch, Lang: language, Train Ac: training accuracy, Test Ac: testing accuracy, F beta: F beta score.

Model Evaluation and Validation

The final model was constructed using one embedding layer, a single LSTM layer and 1 dense output layer. The model was trained with 25 epoch's and achieved a test accuracy of 0.8937 and training validation accuracy of 0.88. Various parameters of the model were modified to improve model predictions. The number of epochs were increased from 10 to 25. Additionally, the number of cell was increase from 100 to 150 and the bidirectional method was applied to the LSTM layer.

Justification

The final model is an improvement over the benchmark. The model accuracy improved while the testing accuracy improved from 86 to 89%. The final model is still not the optimum, as it can be further improved by addition of new LSTM and dense layer to improve the model accuracy.

V. Conclusion

Reflection

When I started researching this project I didn’t have much understanding of RNN. Based on my knowledge of CNN, I research multiple external resources and video tutorials from Coursera Andrew Ng’s course, Stanford NLP course and multiple bogs about RNN/LSTM and NLP. This exercise not only gave me the confidence to start and research a new project but also pointed me to sources of information for deeplearning framework documentation and tutorials.

After researching about possible dataset on Kaggle for fake news classification, started reading about the RNNs and other NLP algorithms. I further learnt about the implementation of RNN LSTM workflow from other blogs for text classification. The first model was built with an embedding layer, LSTM layer and dense output layer using the first 2000 articles. Subsequently, the model was trained with the full data set. The ability to research, write the code and optimize the model from the beginning was a very valuable experience during this project.

The most challenging part of the project has been testing multiple different combinations of the model to find optimum setting for the best trained model. Another difficult part of the project was understanding the LSTM cells mechanism for maintaining memory within sentences. However, with better understanding of the algorithms and settings I will be able to build good models at a faster pace.

Improvement

In the current project the LSTM model for text classification was built with a single LSTM layer and single dense output layer. Additionally, we used only the first 50 words to train and test the model. These two variables may have resulted in limiting the test accuracy to 89%. In the future it would be valuable to build the deeplearning model with 2-4 LSTM layer. Additionally, in the future I will like to include attention in the RNN model as recent research has suggested that this feature can dramatically improve the performance of RNN models. Also, the use of longer word vector for training can improve the RNN algorithm for text classification.

References

1) Text News Classification System using Naïve Bayes Classifiers http://ijoes.vidyapublications.com/paper/Vol13/39-Vol13.pdf
2) Naïve bayes and text classification https://sebastianraschka.com/Articles/2014_naive_bayes_1.html
3) Comment Abuse Classification with Deep Learning: https://web.stanford.edu/class/cs224n/reports/2762092.pdf
4) Recurrent Convolutional Neural Networks for Text Classification: https://www.aaai.org/ocs/index.php/AAAI/AAAI15/paper/download/9745/9552
5) Tweet modeling with LSTM recurrent neural networks for hashtag recommendation https://ieeexplore.ieee.org/document/7727385/
6) Fake news – Wikipedia https://en.wikipedia.org/wiki/Fake_news
7) Facebook to expose Russian fake news pages https://www.bbc.com/news/technology-42096045
8) Fake news stories make real news headlines. https://abcnews.go.com/Technology/fake-news-stories-make-real-news-headlines/story?id=43845383
9) Neural joint model for entity and relation extraction from biomedical text https://bmcbioinformatics.biomedcentral.com/articles/10.1186/s12859-017-1609-9
10) Accuracy, Precision, Recall or F1?. https://towardsdatascience.com/accuracy-precision-recall-or-f1-331fb37c5cb9
11) SKlearn Precision, recall and F-beta score http://scikit-learn.org/stable/modules/generated/sklearn.metrics.precision_recall_fscore_support.html
12) Recurrent Neural Networks Tutorial, Part 1 – Introduction to RNNs http://www.wildml.com/2015/09/recurrent-neural-networks-tutorial-part-1-introduction-to-rnns/
13) LSTM cells http://deeplearning.net/tutorial/lstm.html
14) An Improved Model of Semantic Similarity Based on Lexical Co-Occurrence; Rohde et al 2005
15) Keras embedding layer https://machinelearningmastery.com/use-word-embedding-layers-deep-learning-keras/
16)