

A Deep Model on Hoax Detection Using Feed Forward Neural Network and LSTM

Guntha Venkata Dhanush Kumar

Computer Science and Engineering, Ramaiah Institute of Technology, Bangalore, India.

E-mail: gunthadanush111@gmail.com

Mamatha V Jadhav

Assistant Professor, Computer Science and Engineering, Ramaiah Institute of Technology, Bangalore, India. E-mail: mamsdalvi@msrit.edu

Anvesh Tadiseti

Computer Science and Engineering, Ramaiah Institute of Technology, Bangalore, India.

E-mail: anvesh4t@gmail.com

Kiran

Computer Science and Engineering, Ramaiah Institute of Technology, Bangalore, India.

E-mail: techkiranp@gmail.com

Received July 20, 2020; Accepted September 28, 2020

ISSN: 1735-188X

DOI: 10.14704/WEB/V17I2/WEB17058

Abstract

The topic of hoax news detection on social media has recently pulled in enormous consideration. Social media not taking any credibility for the news being spread in it makes it more difficult to contain the hoax news. The essential counter measure of comparing websites against a list of labeled hoax news sources is inflexible, and so a machine learning approach is desirable. Our project aims to use Neural Networks to detect hoax news directly, based on the text content of news articles. The model concentrates on discovering hoax news origins, based on the many articles originating from it. When a source is spotted as a maker of hoax news, we can predict with high reliability that other articles from that will similarly be hoax news. Focusing on sources augments our article mis categorization resilience, since we at that point have various facts focuses originating from each source.

Keywords

Neural Networks, LSTM, FFNN, RNN, Tensor Flow, Keras.

Introduction

News was always part of our day to day life. Before it used to be from the print and electronic media houses. Things changed a lot now, social media being the current trend in market, we get news from unknown sources in these new platforms without any credibility. Social media have now become a hub for hoax news that's being circulated across the globe.

Any information that is made-up deliberately for misleading people about a person, institution, entity or religion to gain money, to be sensationalist or to create an outrage is termed to be hoax news. With increase in social media platforms and usage of it, hoax news is increasing very rapidly. Hoax news cause lot of disturbances and mental impact in the society. Detecting hoax news is a much-needed thing in the present time. This need helps in reducing the disturbances caused by the hoax news among the people.

Consuming the news on social media is very fast when compared with other things. This is the reason why few deliberate people started using these platforms in order to spread the hoax news. Websites are created which is not an authorized one and start publishing articles in it. These articles are later shared by them in social media platforms as news from credible news website. People without verifying the details of origin of news start circulating the news to all their families and friends causing spread of hoax news. The sources need to be detected in order to contain this issue of hoax news.

The paper aims to bring in a machine learning model which could predict the hoax news. This reduces the burden on people to verify the facts in the news. Different machine learning models are created to check accuracy of each of the models for the hoax news. This model can help overcoming issues that the hoax news can cause as said earlier with a scope to bring down people who could spread them by looking into the sources that frequently spread them. To detect the sources that produce hoax news for gain of money, to spread hatred or to change views of people. Sources once detected to be wide creator of hoax news will be brought down. Bringing in best model to increase the efficiency to detect the hoax news. Creating a pleasant environment in the society by verifying the facts of the articles that are being spread online without any authenticity.

Robust, efficient and conflict free machine learning model to verify the facts in news contents that are being spread by different websites or social media pages. These models can be used to spot the false news circulating online by verifying the claims or stories that they put up on their websites.

Literature Survey

In paper ^[1] they present a thorough survey of finding hoax news on social media, which also includes hoax news portrayals on social speculation and psychology. In data mining point of view, they used the current algorithms. For feature extraction they used : News Content Features - Linguist and Visual based. Social Context Features - It include features from users, posts and network. and for Model Constructions they used: News Content Models-style based, and knowledge based. Social Context Models - Propagation and stance based.

In paper ^[2] they focus on the language patterns followed by deceptors in their language. This research focuses on user reviews and essays but is equally applicable in hoax news detection as well, where the author tries to deceive the readers. For detecting these linguistic patterns, the research uses shallow and deep syntax analysis which use POS (parts-of-speech) tags and Probabilistic Context Free Grammars (PCFG) respectively.

In paper ^[3] they discussed two methods for Fake News detection. The first one is linguistic approach; this discusses the various syntactical and semantically features that are useful in deception detection. It uses deep syntax analysis with context free grammar generation using Stanford parser. The basis of semantic analysis provided in this research is that, the author may use contradictions and omit facts while writing. It also considers that on social media, the authentication of identity of the user posting an article is paramount for the notion of trust.

In paper ^[4] there focus was on the automatically identifying the hoax content in news articles present in websites. They introduced 2 novel datasets for the task of hoax news detection, covering 7 different news domains. They build hoax news detection models by extracting several linguistic features like n-grams, punctuations, psycholinguistic features, readability, syntax etc. They used LIWC to generate these features. Their best models achieved accuracies which are like the human ability to spot fake news.

In paper ^[5] idea is to leverage the three auxiliary information available in the social media regarding the news, publisher and engagers to effectively classify the news content. The tri-relationship is established between news publisher, news and social media users. The framework discussed in this paper is based on semi supervised machine learning classification technique where the three matrices namely User-User Social Relationship, User Credibility and New User Engagement are the basic requirements. Initially, the matrix values are inputted based on the current social media content, as and when the user

relationship and engagement changes, the values are updated parallelly for better classification.

In paper ^[6] study on combination between the feature extraction methods and the classifiers are not done and no proper data mining processes are performed on data. In paper ^[7] they just concentrated on pre-processing methods and didn't bother properly about the data models to be used. In paper ^[8] sources of the news articles were not verified; this brings in a need for a model that could verify the source of news article for checking the hoax news. Paper didn't show psychological view in extracting features to model hoax news and to identify those users who commonly spread hoax news.

Proposed Methodology

A. Dataset

We considered a political dataset with total 20000 rows trained with 16000 rows and validated with 4000 rows provided by Kaggle. The dataset contains id, title, author, text, label columns

id	title	author	text	label
0	House Der	Darrell Luc	House	1
1	FLYNN: Hi	Daniel J. F	Ever get th	0
2	Why the T	Consortiur	Why the	1
3	15 Civilian	Jessica Pul	Videos 15	1
4	Iranian wc	Howard P	Print	1
5	Jackie Ma	Daniel Nu	In these tr	0
6	Life: Life C	nan	Ever	1
7	Benoît F	Alissa J. R	PARIS â€”	0
8	Excerpts F	nan	Donald J. "	0
9	A Back-Ch	Megan Tw	A week be	0
10	Obamaâ€”	Aaron Klei	Organizing	0
11	BBC Come	Chris Tom	The BBC p	0
12	Russian Re	Amando F	The	1
13	US Official	Jason Ditz	Clinton	1
14	Re: Yes, Th	AnotherAr	Yes,	
BART SIMPSONSON				
Hey	itâ€™s jus	channels	and programs	fellating them da
Itâ€™s not	I imagine	oil compa	difficult to know	who to trust o
In any soc	most people	do nothing.	Itâ€™s up to the	minority to
If I read the	article correctly	the government	is targeting conserv	
The DNC is	stupid and	but these j@ck@sses	ramp it up to 11.)	Ta
I almost pi	which wa	especially	1	

Figure 1 View of DataSet

B. Preprocessing

In this part we first removed stop words, special characters and punctuation and this gives the list of words which is given as input to the Doc2vec model to get the vectors of size 300 which is used as input to neural Network model.

C. Implementation of Algorithm

In this project we have implemented three models among these models two are feed forward neural network models, one utilizing TensorFlow and one utilizing Keras. Our neural system executions utilize three hidden layers. In the TensorFlow execution all layers have 300 neurons each and in the Keras usage we utilized layers of size 256, 256, and 80, mixed with dropout layers to abstain from overfitting. For our initiation work, we picked the Rectified Linear Unit (ReLU), which has been found to perform well in NLP applications.

What's more, one progressively model is LSTM it was proposed by Hochreiter and Schmidhuber. It is acceptable at ordering serialized objects since it will specifically memorize the previous input info and utilize that, along with the present contribution, to make expectation. The news (content) in our concern is inherently serialized. The request for the words conveys the significant data of the sentence. In this way, the LSTM model suits for our concern. Since the request for the words is significant for the LSTM unit, we can't utilize the Doc2Vec for preprocessing in light of the fact that it will move the whole document into one vector and lose the request data. To forestall that, we utilize the word embedding. We first clean the content data by expelling all characters which are not letters nor numbers. At that point we check the recurrence of each word showed up in our preparation dataset to discover 5000 most regular words and give everyone a unique integer ID. For instance, the most widely recognized word will have ID 0, and the second most regular one will have 1, and so forth. After that we supplant every basic word with its allocated ID and erase every phenomenal word. Notice that the 5000 most basic words spread a large portion of the content, so we just lose little data however move the string to a rundown of numbers.

Since the LSTM unit requires a fixed information vector length, we shorten the list longer than 500 numbers since the greater part of the news is longer than 500. At that point for those rundowns shorter than 500 words, we pad 0's toward the beginning of the list. We likewise erase the data with just a couple of words since they don't convey enough data for preparing. By doing this, we move the first content string to a fixed length integer vector while saving the words request data. At long last, we use word embedding to move each word ID to a 32-measurement vector. The word embedding will prepare each word vector dependent on word closeness. On the off chance that two words as often as possible show up together in the content, they are believed to be progressively comparable and the separation of their relating vectors is small.

The pre-processing moves every news in crude content into a fixed size matrix. At that point we feed the processed training data into the LSTM unit to prepare the model. The LSTM is yet a neural network. In any case, not the same as the completely associated neural network, it has cycle in the neuron connections. Along these lines, the previous state (or memory) of the LSTM will assume a job in new prediction.

Design Architecture

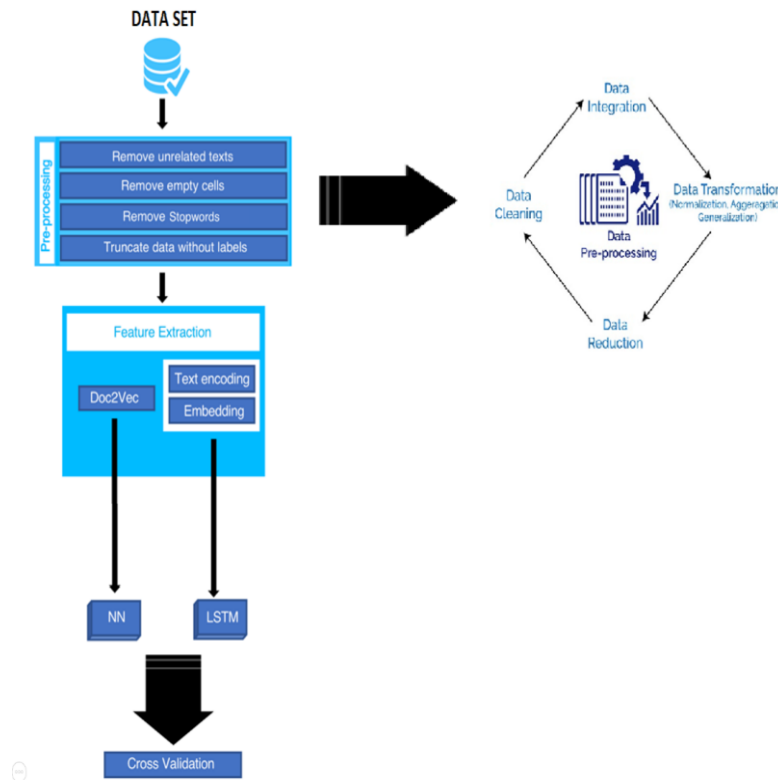


Figure 2 Design Architecture

Performance Comparison

Table 1 Performance comparison of models

Model	Accuracy	Precision	Recall	F1 score
FFNN(Using TensorFlow)	82.63%	80.7181%	86.4262%	83.4747%
FFNN(Using Keras)	92.59%	92.5211%	92.6329%	92.5769%
RNN(LSTM)	94.85%	95.5209%	94.1459%	94.8284%

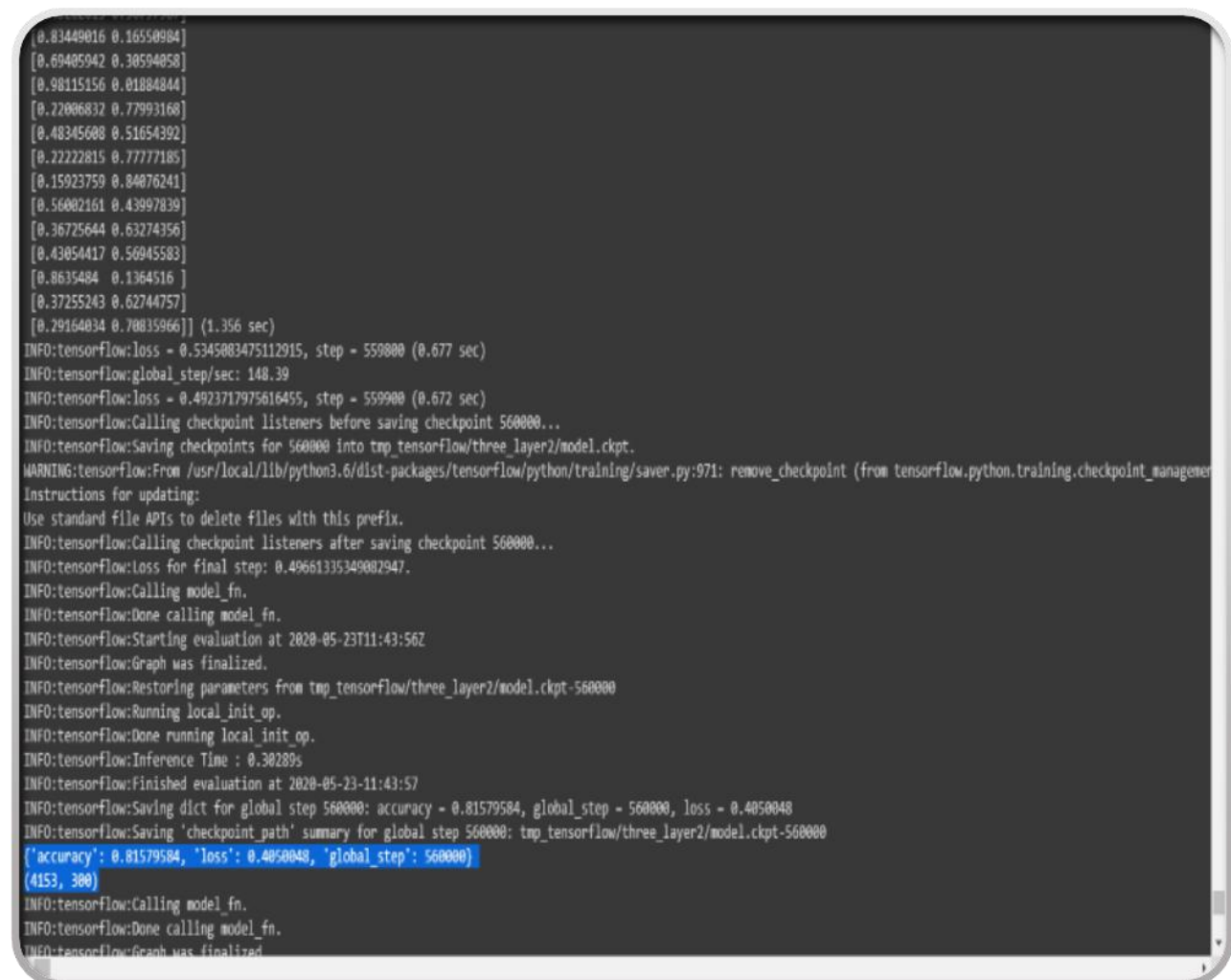
From the above table we can conclude that FFNN using TensorFlow gives an accuracy of 82.63%, FFNN using Keras gives an accuracy of 92.59%, RRN (LSTM) gives an accuracy of 94.85%.

From the above table we can conclude that FFNN using TensorFlow gives a precision of 80.7181%, FFNN using Keras gives an accuracy of 92.5211%, RRN (LSTM) gives an accuracy of 95.5209%.

From the above table we can conclude that FFNN using TensorFlow gives a Recall of 86.4262%, FFNN using Keras gives an accuracy of 92.6329%, RRN (LSTM) gives an accuracy of 94.1459%.

From the above table we can conclude that FFNN using TensorFlow gives a F1 Score of 83.4747%, FFNN using Keras gives an accuracy of 92.5769%, RRN (LSTM) gives an accuracy of 94.8284%.

Results Snapshots of Different Models



```
[0.83449016 0.16550984]
[0.69405942 0.30594058]
[0.98115156 0.01884844]
[0.22006832 0.77993168]
[0.48345608 0.51654392]
[0.22222815 0.77777185]
[0.15923759 0.84076241]
[0.56002161 0.43997839]
[0.36725644 0.63274356]
[0.43054417 0.56945583]
[0.8635484 0.1364516 ]
[0.37255243 0.62744757]
[0.29164034 0.70835966]] (1.356 sec)
INFO:tensorflow:loss = 0.5345083475112915, step = 559000 (0.677 sec)
INFO:tensorflow:global_step/sec: 148.39
INFO:tensorflow:loss = 0.4923717975616455, step = 559900 (0.672 sec)
INFO:tensorflow:Calling checkpoint listeners before saving checkpoint 560000...
INFO:tensorflow:Saving checkpoints for 560000 into tmp_tensorflow/three_layer2/model.ckpt.
WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/tensorflow/python/training/saver.py:971: remove_checkpoint (from tensorflow.python.training.checkpoint_manager)
Instructions for updating:
Use standard file APIs to delete files with this prefix.
INFO:tensorflow:Calling checkpoint listeners after saving checkpoint 560000...
INFO:tensorflow:Loss for final step: 0.49661335349002947.
INFO:tensorflow:Calling model_fn.
INFO:tensorflow:Done calling model_fn.
INFO:tensorflow:Starting evaluation at 2020-05-23T11:43:56Z
INFO:tensorflow:Graph was finalized.
INFO:tensorflow:Restoring parameters from tmp_tensorflow/three_layer2/model.ckpt-560000
INFO:tensorflow:Running local_init_op.
INFO:tensorflow:Done running local_init_op.
INFO:tensorflow:Inference Time : 0.30289s
INFO:tensorflow:Finished evaluation at 2020-05-23-11:43:57
INFO:tensorflow:Saving dict for global step 560000: accuracy = 0.81579584, global_step = 560000, loss = 0.4050048
INFO:tensorflow:Saving 'checkpoint_path' summary for global step 560000: tmp_tensorflow/three_layer2/model.ckpt-560000
('accuracy': 0.81579584, 'loss': 0.4050048, 'global_step': 560000)
(4153, 300)
INFO:tensorflow:Calling model_fn.
INFO:tensorflow:Done calling model_fn.
INFO:tensorflow:Graph was finalized
```

Figure 3 FFNN using TensorFlow (Train and test)


```

0.08208758 -0.2035051 0.12189708 0.1889588 -0.08359861 -0.08802811
-0.07389537 0.02955014 -0.00786195 0.09626734 0.1587693 0.09122135
-0.0756108 0.27635043 0.00524466 -0.04882846 -0.0023138 0.26373258
-0.04158019 -0.24188341 0.27183026 -0.02893004 -0.11267031 0.1603439
-0.27455476 0.10679983 -0.15602277 0.1671894 -0.04577224 0.3381751
0.10563767 0.15048817 -0.27194735 0.15766637 -0.30987513 0.13913795
0.05077693 -0.04495786 -0.13516712 -0.02500948 -0.04545759 0.00434943
0.4230854 0.05101424 0.04221608 -0.3096298 0.09450325 -0.00514079
-0.00105028 0.36265388 -0.0667706 0.3021879 0.11123986 0.3049475
-0.1046055 -0.05701302 -0.09732889 0.20759259 0.21797659 -0.23347558
-0.04312011 0.00306716 -0.13348219 0.12354254 -0.23263752 0.06288045
0.07561849 0.0858717 -0.05111024 -0.08030237 0.23129043 0.10634655
-0.07930117 0.18049777 0.2237848 -0.22903259 -0.10367612 -0.31587988
-0.28912133 0.04119211 -0.02312008 0.04023573 0.15157044 -0.00524733
0.10115085 0.19645336 0.19841485 -0.01299158 -0.23030667 0.1185018
0.19580689 0.2323617 -0.32032767 0.25765753 -0.20895766 -0.3855473
0.41899663 0.0090405 0.05211975 0.3151814 0.11124591 0.18598837
-0.20302921 0.07920652 0.30042124 0.02719825 -0.13176674 0.01204725
-0.04742314 -0.20248042 0.17931554 0.3034040 0.1340040 0.2031076
-0.02231777 -0.02410401 0.12914374 0.05004643 0.15272513 -0.05184651
0.00916974 -0.18000314 0.04281269 -0.15190138 -0.295529 -0.05784978
0.27280506 0.01314592 -0.03764809 0.03738618 -0.23118712 -0.05063118
-0.20974712 -0.06862751 -0.07416627 0.10586319 0.12749863 -0.22708814
-0.06913027 0.03281216 0.24068823 0.08435991 0.2925664 -0.00846427
-0.04660927 0.02570866 -0.01689469 -0.08878994 0.27521926 0.20338857
-0.3151397 0.10240685 0.12758772 -0.06712694 -0.24728687 -0.0043462
-0.02984992 0.21236733 0.25232527 -0.05929695 0.4440304 0.31141436]
<class 'numpy.ndarray'>
float32
float64
(1, 300)
<generator object Estimator.predict at 0x7fca4a292570>
INFO:tensorflow:Calling model_fn.
INFO:tensorflow:Done calling model_fn.
INFO:tensorflow:Graph was finalized.
INFO:tensorflow:Restoring parameters from tmp_tensorflow/three_layer2/model.ckpt-560000
INFO:tensorflow:Running local_init op.
INFO:tensorflow:Done running local_init op.
{'classes': 0, 'probabilities': array([0.6017357, 0.3982643])}
Select the choice
1- Train and Test
2- Test with Custom input
3- Exit
3

```

Figure 4 FFNN using TensorFlow (Test with Custom Input)

```

Using TensorFlow backend.
2020-05-23 11:50:52.367826: I tensorflow/stream_executor/platform/default/dso_loader.cc:44] Successfully opened dynamic library libcudart.so.10.1
Select the choice
1- Train and Test
2- Test with Custom input
3- Exit
1
LabeledSentence(['house', 'den', 'aide', 'even', 'see', 'comey', 'letter', 'jason', 'chaffetz', 'tweeted', 'darrell', 'lucus', 'october', '30', '2016', 'subscribe', 'jason',
(20761,)
20761
16608
4153
2020-05-23 11:57:28.276475: I tensorflow/stream_executor/platform/default/dso_loader.cc:44] Successfully opened dynamic library libcuda.so.1
2020-05-23 11:57:28.278333: E tensorflow/stream_executor/cuda/cuda_driver.cc:313] failed call to cuInit: CUDA_ERROR_NO_DEVICE: no CUDA-capable device is detected
2020-05-23 11:57:28.278374: I tensorflow/stream_executor/cuda/cuda_diagnostics.cc:156] kernel driver does not appear to be running on this host (c0da52195235): /proc/driver/nv
2020-05-23 11:57:28.284302: I tensorflow/core/platform/profile_utils/cpu_utils.cc:102] CPU Frequency: 2200000000 Hz
2020-05-23 11:57:28.284539: I tensorflow/compiler/xla/service/service.cc:168] XLA service 0x21ef2c0 initialized for platform Host (this does not guarantee that XLA will be use
2020-05-23 11:57:28.284578: I tensorflow/compiler/xla/service/service.cc:176] StreamExecutor device (0): Host, Default Version
Model: "sequential_1"

Layer (type)                Output Shape                Param #
-----
dense_1 (Dense)              (None, 256)                 77056
dropout_1 (Dropout)          (None, 256)                 0
dense_2 (Dense)              (None, 256)                 65792
dropout_2 (Dropout)          (None, 256)                 0
dense_3 (Dense)              (None, 80)                  20560
dense_4 (Dense)              (None, 2)                   162
-----
Total params: 163,570
Trainable params: 163,570
Non-trainable params: 0

Epoch 1/20
386/13286 [=====] - 1s 77ms/step - loss: 0.5249 - accuracy: 0.7254

```

Figure 5 FFNN using Keras (Train and test)


```

2.67287988e-01 -2.18899397e-02 2.89923001e-02 2.88338357e-01
2.15359107e-01 -4.24898040e-02 5.24106576e-02 1.15535310e-01
-2.20825884e-01 5.98332379e-03 6.35504428e-02 2.88121470e-01
3.83783169e-02 1.77114248e-01 1.17357455e-01 -3.46237384e-02
1.81907654e-01 -2.05357447e-01 2.78505594e-01 1.36328757e-01
-1.47911459e-01 1.24635756e-01 2.58621238e-02 2.62308002e-01
-3.39045793e-01 -8.50492269e-02 1.30366027e-01 3.84200886e-02
4.33439344e-01 2.06287310e-01 3.15796971e-01 2.40530312e-01
-1.95347384e-01 -3.55535737e-02 -1.14041783e-01 2.86672208e-02
-2.29939163e-01 -1.03813291e-01 -9.31753516e-02 1.96564198e-01
-1.84725106e-01 -2.88229555e-01 2.91370321e-02 -1.86984152e-01
1.39676675e-01 3.60006481e-01 -2.7152643e-02 -4.99401428e-02
-1.22028030e-01 -1.34797618e-01 8.71327072e-02 -3.46976593e-02
-2.99681686e-02 7.12653473e-02 2.23966941e-01 -2.32180238e-01
2.37077564e-01 1.72072947e-02 5.07321320e-02 6.89961109e-03
-1.63571894e-01 -4.02658165e-01 -3.08956325e-01 1.15872644e-01
3.17716859e-02 4.99057800e-01 -1.20800594e-02 1.91465750e-01
2.76274681e-02 -3.19931537e-01 2.91173570e-02 -2.07149446e-01
1.35549963e-01 4.42895830e-01 -2.77816892e-01 -3.46252285e-02
-1.87592134e-01 2.36954734e-01 -4.71093729e-02 7.66581669e-03
-8.73440579e-02 -1.38872205e-01 -2.71523185e-02 -1.71449944e-01
1.11721113e-01 1.24180451e-01 6.42266124e-02 -5.83349243e-02
-1.85270324e-01 -2.09823381e-02 1.71851031e-02 -7.54300365e-03
-8.13847855e-02 -1.04976885e-01 -1.40792102e-01 4.47887816e-02
2.89545953e-02 -1.28659904e-01 -3.69815038e-02 1.92485656e-01
-2.64645875e-01 3.13107818e-01 -9.45311487e-02 2.20898148e-02
-1.93151589e-02 -1.50294170e-01 1.86587363e-01 -7.22228549e-03
-1.91593287e-03 1.13499172e-01 2.17460226e-02 -1.66204610e-01
9.77127478e-02 1.51477084e-01 3.54632199e-01 2.11323977e-01
3.60816002e-01 1.83305323e-01 1.85905978e-01 -8.36397931e-02
1.92717060e-01 1.00042768e-01 -1.78358909e-02 -2.00679898e-01
1.03373311e-01 1.40955093e-01 -1.06272955e-02 1.59574091e-01
1.03322136e-01 0.10258315e-03 2.31030206e-01 -1.21162571e-01
-2.21371427e-01 1.54435024e-01 1.36420220e-01 -1.39709243e-02
1.30540389e-01 -3.30586568e-04 -1.89947880e-01 3.03172320e-01
1.48077175e-01 1.66860402e-01 1.93299785e-01 -1.10717520e-01]
(1, 300)
[[0.7160438 0.28395626]]
[[0.7160438 0.28395626]]
Select the choice
1- Train and Test
2- Test with Custom Input
3- Exit
3

```

Figure 6 FFNN using Keras(test with Custom Input)

```

3- Exit
1
2020-05-24 03:28:16.337416: I tensorflow/stream_executor/platform/default/dso_loader.cc:44] Successfully opened dynamic library libcuda.so.1
2020-05-24 03:28:16.340310: E tensorflow/stream_executor/cuda/cuda_driver.cc:313] failed call to cuInit: CUDA_ERROR_NO_DEVICE: no CUDA-capable device is detected
2020-05-24 03:28:16.340385: I tensorflow/stream_executor/cuda/cuda_diagnostics.cc:156] kernel driver does not appear to be running on this host (4691a6e22b96):
2020-05-24 03:28:16.351175: I tensorflow/core/platform/profile_utils/cpu_utils.cc:102] CPU Frequency: 2300000000 Hz
2020-05-24 03:28:16.351528: I tensorflow/compiler/xla/service/service.cc:168] XLA service 0x2c93b80 initialized for platform Host (this does not guarantee that
2020-05-24 03:28:16.351574: I tensorflow/compiler/xla/service/service.cc:176] StreamExecutor device (0): Host, Default Version
Model: "sequential_1"

Layer (type)                 Output Shape                 Param #
-----
embedding_1 (Embedding)      (None, 500, 32)             160064
lstm_1 (LSTM)                 (None, 100)                  53200
dense_1 (Dense)              (None, 1)                    101
-----
Total params: 213,365
Trainable params: 213,365
Non-trainable params: 0

None
/usr/local/lib/python3.6/dist-packages/tensorflow/python/framework/indexed_slices.py:434: UserWarning: Converting sparse IndexedSlices to a dense Tensor of unknown shape.
"Converting sparse IndexedSlices to a dense Tensor of unknown shape."
Train on 16030 samples, validate on 4153 samples
Epoch 1/5
16030/16030 [-----] - 177s 11ms/step - loss: 0.3399 - accuracy: 0.8637 - val_loss: 0.2698 - val_accuracy: 0.9080
Epoch 2/5
16030/16030 [-----] - 178s 11ms/step - loss: 0.1774 - accuracy: 0.9414 - val_loss: 0.1723 - val_accuracy: 0.9398
Epoch 3/5
16030/16030 [-----] - 175s 11ms/step - loss: 0.1187 - accuracy: 0.9592 - val_loss: 0.1331 - val_accuracy: 0.9518
Epoch 4/5

```

Figure 7 LSTM (train and test)

```
usr/local/lib/python3.6/dist-packages/tensorflow/python/framework/indexed_slices.py:434: UserWarning: Converting sparse IndexedSlices to a dense Tensor of unknown shape.
"Converting sparse IndexedSlices to a dense Tensor of unknown shape. "
Train on 16030 samples, validate on 4153 samples
Epoch 1/5
16030/16030 [=====] - 217s 14ms/step - loss: 0.4946 - accuracy: 0.8072 - val_loss: 0.5357 - val_accuracy: 0.6805
Epoch 2/5
16030/16030 [=====] - 191s 12ms/step - loss: 0.4761 - accuracy: 0.7730 - val_loss: 0.3253 - val_accuracy: 0.8627
Epoch 3/5
16030/16030 [=====] - 173s 11ms/step - loss: 0.2793 - accuracy: 0.8884 - val_loss: 0.3180 - val_accuracy: 0.8608
Epoch 4/5
16030/16030 [=====] - 174s 11ms/step - loss: 0.2101 - accuracy: 0.9201 - val_loss: 0.2435 - val_accuracy: 0.9018
Epoch 5/5
16030/16030 [=====] - 181s 11ms/step - loss: 0.1317 - accuracy: 0.9551 - val_loss: 0.1601 - val_accuracy: 0.9461
Accuracy= 94.61%
Precision: 0.939924
Recall: 0.953455
F1 score: 0.946641
/usr/local/lib/python3.6/dist-packages/sklearn/utils/deprecation.py:87: FutureWarning: Function plot_confusion_matrix is deprecated; This will be removed in v0
warnings.warn(msg, category=FutureWarning)
Select the choice
1- Train and Test
2- Test with Custom input
3- Exit
2
Enter the news that want to be checked:
President Donald Trump urged members of the NATO alliance to start paying their fair share Thursday in Brussels, pointing out that their failures were hurting
[['president', 'donald', 'trump', 'urged', 'members', 'nato', 'alliance', 'start', 'paying', 'fair', 'share', 'thursday', 'brussels', 'pointing', 'failures', '
[12, 59, 3, 2171, 171, 912, 1948, 396, 1636, 1579, 419, 335, 3403, 3003, 23, 3972, 912, 171, 239, 738, 4387, 1579, 419, 1040, 328, 3, 1, 3003, 1805, 1082, 171
[0]]
[0.29965916]]
Select the choice
1- Train and Test
2- Test with Custom input
```

Figure 8 LSTM (test with custom Input)

Conclusion and Scope for Future Work

A total, production quality classifier will consolidate a wide range of highlights past the vectors comparing to the words in the content. For hoax news discovery, we can include as highlights the source of the news, including any related URLs, the subject (e.g., science, legislative issues, sports, and so forth.), distributing medium (blog, print, online life), geographic region of origin, publication year, as well as linguistic features not exploited in this exercise use of capitalization, fraction of words that are proper nouns (using gazetteers), and others.

Besides, we can likewise total the all-around performed classifiers to accomplish better precision. For instance, utilizing bootstrap totaling for the Neural Network, LSTM to show signs of better prediction result.

A similar work is searching the news on the Internet and compare the search results with the original news. Since the item is typically dependable, this technique ought to be increasingly precise, yet in addition includes natural language understanding in light of

the fact that the indexed lists won't be actually equivalent to the original news. Along these lines, we should look at the significance of two contents and decide whether they mean something very similar.

References

- Shu, K., Sliva, A., Wang, S., Tang, J., & Liu, H. (2017). Fake news detection on social media: A data mining perspective. *ACM SIGKDD explorations newsletter*, 19(1), 22-36.
- Feng, S., Banerjee, R., & Choi, Y. (2012). Syntactic stylometry for deception detection. *In Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (Short Papers)*, 2, 171-175.
- Conroy, N.K., Rubin, V.L., & Chen, Y. (2015). Automatic deception detection: Methods for finding fake news. *Proceedings of the Association for Information Science and Technology*, 52(1), 1-4.
- Pérez-Rosas, V., Kleinberg, B., Lefevre, A., & Mihalcea, R. Automatic detection of fake news 2017. *arXiv preprint arXiv:1708.07104*.
- Shu, K., Wang, S., & Liu, H. Exploiting tri-relationship for fake news detection 2017. *arXiv preprint arXiv:1712.07709*, 8.
- Al Asaad, B., & Erascu, M. (2018). A tool for fake news detection. *In IEEE 20th International Symposium on Symbolic and Numeric Algorithms for Scientific Computing (SYNASC)*, 379-386.
- Kotteti, C.M.M., Dong, X., Li, N., & Qian, L. (2018). Fake news detection enhancement with data imputation. *IEEE 16th Intl Conf on Dependable, Autonomic and Secure Computing, 16th International Conference on Pervasive Intelligence and Computing, 4th International Conference on Big Data Intelligence and Computing and Cyber Science and Technology Congress (DASC/PiCom/DataCom/CyberSciTech)*, 187-192.
- Parikh, S.B., & Atrey, P.K. (2018). Media-rich fake news detection: A survey. *In IEEE Conference on Multimedia Information Processing and Retrieval (MIPR)*, 436-441.