VIT®

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)

Project Title:

Effective Fake News detection with Neural Networks

Team Members:

Name	Registration Number
Shashvat Adhaduk	19BIT0176
Eashaa Shiv Shankar Singh Kushwah	19BIT0206
Shourya Singh	19BIT0208
Kunj Patel	19BIT0256

1. Abstract

Fake news for different economic and political reasons has been surfacing in enormous numbers and spreading in the internet world in recent years, thanks to the explosive development of online social networks. Online social network members may quickly become infected by online false news by using misleading phrases, which has already had a huge impact on the offline culture. Identifying false news in real time is an essential objective in enhancing the credibility of information in online social networks. The goal of this project is to look at the ideas, approaches, and algorithms for detecting false news pieces, makers, and subjects on online social networks, as well as assessing their effectiveness.

2. Introduction

Fake news is a type of yellow press that deliberately spreads disinformation or hoaxes through traditional print news media as well as contemporary internet social media. Fake news for various commercial and political goals has been surfacing in big numbers and spreading in the internet world in recent years as a result of the blooming development of online social networks. Online social network members can quickly become infected by online fake news by using deceptive phrases, which has already had a huge impact on the offline culture.

Fake news differs significantly from traditional suspicious information in that spams typically exist in personal emails or specific review websites and only have a local impact on a small number of audiences, whereas the impact of fake news in online social networks can be enormous due to the massive user numbers worldwide, which is boosted further by extensive information sharing and propagation among these users.

While we can understand why detecting fake news items is important, identifying the fake news authors and subjects proves even more crucial, as it will help erase a big number of fake news stories from their online social media roots. As a result, identifying fake news quickly and minimizing the influence it has on the lives of people exposed to it is a key goal in enhancing the reliability of information in online social networks. Thus with this understanding, we can comprehend the reasoning behind the aims of our project.

3. Related work

[1] Deep neural approach to Fake-News identification

The current solutions lack at identifying the authenticity of a well-written article is its shortcoming. Thus, this paper presents a Live Data Mining section along with deep learning models to eliminate the issue. The mining stage finds secondary features in a post such as domain name, authors and their credibility, etc. For data mining from web/web scraping Beautifulsoup HTML Parser using Python was used. While Feed Forward and LSTM were chosen as the Neural Network of the choice. After using the George McIntire Dataset, it was concluded that LSTM performed better than FNN, as with data mining it offered an accuracy of 91.32% and Recall of 94.21%, far surpassing the ones offered by FNN. Better results were observed with word2vec, word vector model, as compared to GloVe.

[2] Optimization and improvement of fake news detection using deep learning approach for the societal benefit

Fake news is regarded as an epidemic and considered a hotly debated current topic. It tends to spread faster and is unrecognized. The goal is to design a model which minimizes fake news by categorizing it as true or false. For Data-Preprocessing NLTK is used to remove stopwords.CountVectorizer, a Python tool is used to convert a given text into a vector-based on the frequency (count) of each word in the full document, and this process is termed tokenization.While GloVe is used to generate representations for those word vectors. A Sequential Model is designed with the help of LSTM and offers an accuracy of 99.88%

[3] A Novel Approach Towards Fake News Detection: Deep Learning Augmented with Textual Entailment Features

The proliferation of multiple data produced by us online has to be checked for validity. The approach taken is to compare one article to other publications and see if similar articles exist. A supervised ML model based on SVM and MLP is designed to compare articles based on grammatical terms, such as Hyponyms, Antonyms, etc. Two more approaches based on DL are introduced as well, applying Universal Sentence Encoder (USE) in one of them while in the other approach USE is used with ML features. Using a proprietary scoring system known as FNC (Fake News Challenge), the ML model evaluates at 72.13 while the DL approaches each have a score of 76.9 and 82.54, respectively. Thus, concluding that the 2nd DL approach is the best amongst all.

[4] Fake News Detection Using Machine Learning Ensemble Methods

Since we make decision based on the information we consume, a rumor or a fake news can have disastrous effects. Most of the current solutions focus on a particular domain of News, mainly Politics, thus Algorithms do not achieving optimal results when it comes to different domains as they have unique textual structure. The approach to find a solution for this is to train an ensemble

of ML algorithms based on textual properties. Various algorithms are compared for finding an optimal solution. The average accuracy obtained by ensemble learners is 97.67% on the DS1 dataset, whereas the corresponding average for individual learners is 95.25%, Decision Tree and Perez LSVM performing the best among the bunch.

[5] DeepFakE:
Improving fake news detection using tensor decomposition-based deep neural network Due to the sheer ease of accessing and sharing data due to which fake news has multiplied, necessitating a review of the legitimacy and authenticity of news stories posted on social media. This paper proposes a deep neural network model (DeepFakE) for modeling combined matrix-tensor factorization results from both news content and social context as echo-chambers, for fake news detection. The proposed model (DeepFakE) resulted with a validation accuracy of 85.86 percent and 88.64 percent ,precision value of 0.8333 and 0.8210 with Buzzfeed and PolitiFact dataset respectively .

[6] Fake News Detection Using a Blend of Neural Networks: An Application of Deep Learning
False news and its effects have the ability to affect a wide range of entities, but no commercially viable solution exists .This research presents a model for predicting and detecting false news based on feature extraction utilizing deep learning techniques i.e. convolutional and recurrent neural networks, with the former offering faster feature extraction whereas the latter assisting the model through its sequential and memory-holding properties trained on dataset chosen from a Kaggle. With a precision of 97.21 percent and an accuracy of 99.54 percent, this model meets the benchmark, with the addition of word embeddings complementing the model together.

[7] Fake News Detection with Deep Diffusive Network Model
Fake news for various commercial and political purposes has been appearing in large numbers making it is necessary to identify fake news .Here research presents the framework FakeDetector, which uses the gated diffusive unit model along with HFLU feature learning, to infer the credibility of news using both explicit and latent textual characteristics and relational linkages.The credibility inference problem is decomposed into a binary class classification and a multi-class classification problem where FakeDetector displayed best results when compared to other models, evaluated with measures such as Accuracy, Precision, Recall, Macro F1, and Macro Precision used to assess performance.

[8] FNDNet –
A deep convolutional neural network for fake news detection The spread of fake news has become a big problem to numerous industries and agencies ,this has eroded faith in the media, leaving readers perplexed.The suggested model in this study is FNDNet, a deep convolutional neural network that uses GloVe as a pre-trained word embedding to detect fake news. For classification, several machine learning techniques, as well as deep learning methods, are used.

FNDNet gives trailblazing results for predicting fake news with an accuracy of 98.36%, which has been validated using several performance evaluation parameters such as, precision, false positive rate ,recall, true negative rate, and so on.

[9] A Smart System for Fake News Detection Using Machine Learning:
It is harmful for society to believe in rumors and fake news; hence the proposed system helps to find the authenticity of the news. If the news is not real, then the user is suggested with the relevant news article. Here the proposed model is a mix of Naïve Bayes classifier, Support Vector Machines, and semantic investigation. The aforementioned model achieved an accuracy of 93.50% when implemented against four existing approaches.

[10] Truth of Varying Shades:
Analyzing Language in Fake News and Political Fact-Checking: The paper presents an analytic study on the language of news & media for political fact-checking and fake news detection. Aside from linguistic comparisons made in the study, ML algorithms like Naïve Bayes, MaxEnt and LSTM were utilized to check the truthfulness of news articles. LSTM outperformed the other two (based on macro averaged F1 scores) when text was used as input, but the other two improved substantially when adding LIWC features.

[11] Some Like it Hoax: Automated Fake News Detection in Social Networks:
The research tries to demonstrate that the number of people who engage with news items on social media platforms may be used to forecast whether or not the posts are fake. They used two approaches for this purpose i.e., Logistic regression and Boolean label crowdsourcing (BLC). Both approaches provided good performance, with especially BLC giving accuracy above 99% even when trained only over sets of posts containing only 0.5% of the dataset.

[12] CSI: A Hybrid Deep Model for Fake News Detection:
In this work a study was performed for the timely problem of fake news detection. Where a CSI model composed of three modules for ‘capturing’, ‘scoring’ and ‘integrating’ was used. These different modules allowed a prediction to be made separately for users and articles. A comparison was made over 2 datasets, namely Twitter and Weibo where high detection accuracies of 0.892 & 0.953, as well as F-scores of 0.894 &0.954 were obtained respectively.

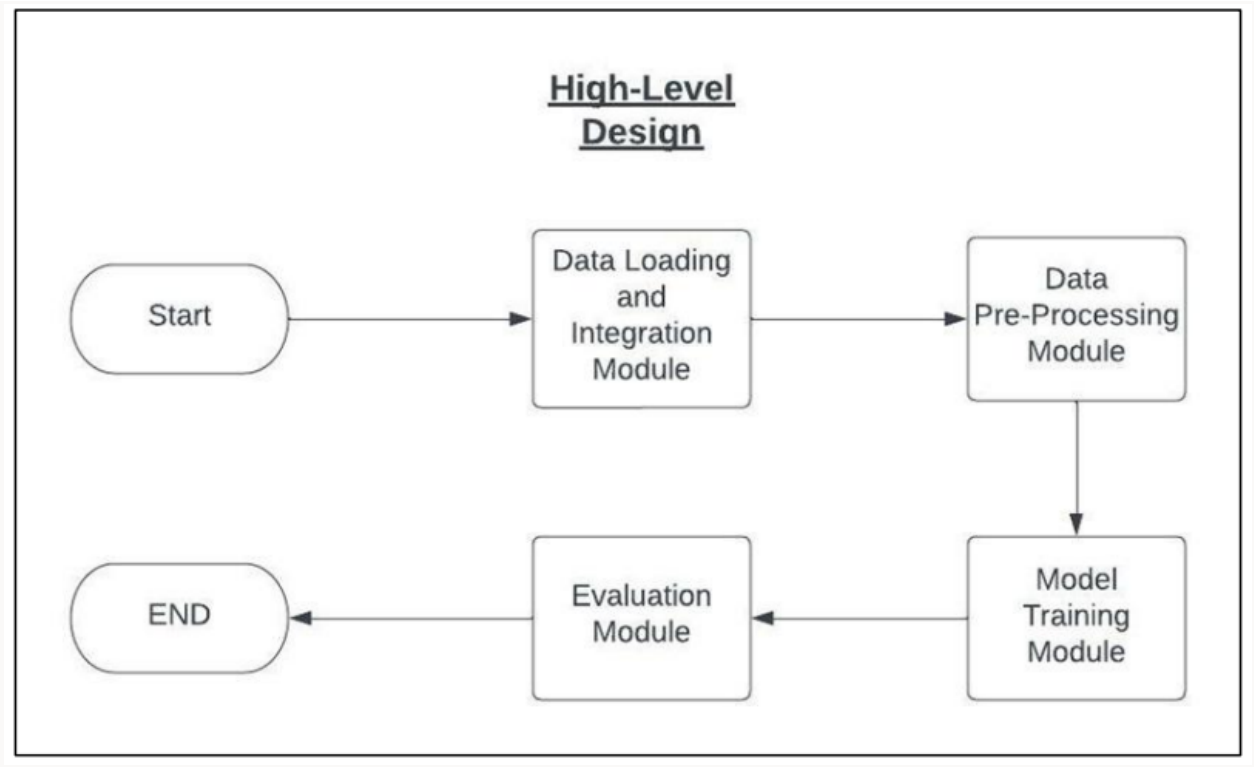
[13] Fake News Detection using Bi-directional LSTM-Recurrent Neural Network Convolutional Neural Networks (CNN) and Recurrent Neural Networks (RNN) are two common deep learning models that can discover complicated patterns in textual data to identify fake news. But these types of models are not suitable for analyzing huge length data, as well as unidirectional analysis. In this paper, a Long Short-Term Memory (LSTM) neural network is used for the purpose of analyzing variable length sequential data. Especially here a Bi-directional LSTM model is used to allow analyzing a sequence from both front-to-back as well as back-to-front.

[14] Fake News Detection using Deep Learning
The goal of this research is to use natural language processing (NLP) techniques for text analytics and to train deep learning models to detect false news based on the title or content of the news. The solution proposed via this paper aims at eradicating all the fake news.. Receiving false information is an unpleasant experience for users from an untrustworthy source. Text preparation is important for NLP approaches, regular expressions, tokenization, and so on. Before, stop words were removed and lemmatization was applied. N-gram vectors or sequence vectors are created by vectorizing them. Using the words inverse document frequency and frequency (TF-IDF) or one-shot encoding, as the case may be. The models' results suggest that models trained with news content can achieve better results while sacrificing computation time, whereas models trained with news titles require less computation time to obtain decent results. Additionally, models fed with N-gram vectors perform marginally better than models fed with sequence vectors in terms of overall performance.

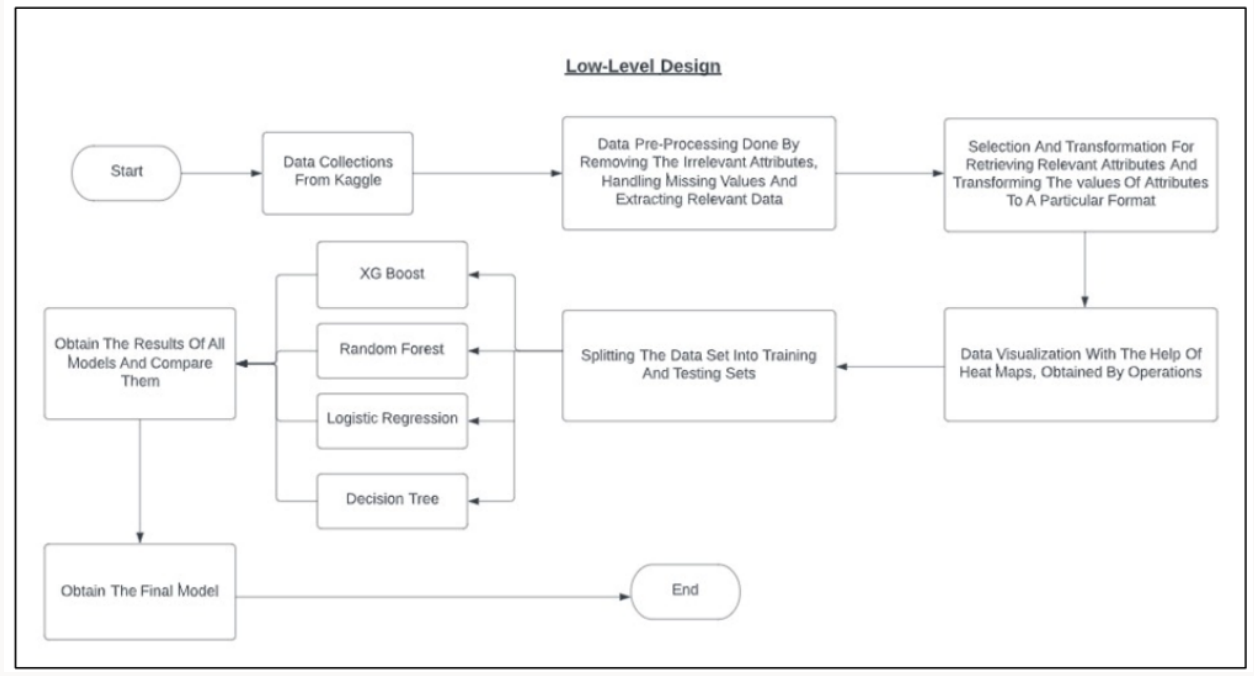
[15] TI-CNN: Convolutional Neural Networks for Fake News Detection
They propose to investigate the problem of "fake news detection" in this research. Because pure model-based fact-checking for news is still an open problem with few available models that can be applied to tackle the problem, detecting false news automatically is incredibly difficult. Many valuable explicit elements are determined from both the full texts and visuals utilized in the fake news after a thorough study of the data. Aside from the visible features, there are some underlying patterns in the texts and pictures used in false news that can be caught using a collection of latent features using the model's numerous convolutional layers. They present TICNN, a unified model that can incorporate text and picture information with the relevant explicit and latent features, in this study. The suggested model provides a high degree of expandability, allowing it to easily absorb more aspects of news. Furthermore, the convolutional neural network allows the model to see all of the input at once and can be trained considerably quicker than the LSTM and many other RNN models.

[16] DEAP-FAKED: Knowledge Graph based Approach for Fake News Detection
Their approach combines NLP and the GNN techniques. A variety of such encodings gives our detector an additional benefit. They also eliminate bias, such as the origin of the articles, as part of the dataset pre-processing, which could affect the models' performance. DEAP-FAKED achieves an F1-score of 88 percent and 78 percent for two data sets, respectively, a 21 percent and 3 percent improvement, demonstrating the usefulness of the strategy.

4. Proposed methodology



Low-Level Architecture Design for the implementation of ML algorithms:



Implementation of the Modules:

LOGISTIC REGRESSION

● Logistic regression is one of the most popular Machine Learning algorithms, which comes under the Supervised Learning technique. It is used for predicting the categorical dependent variable using a given set of independent variables.

● Logistic regression predicts the output of a categorical dependent variable. Therefore, the outcome must be a categorical or discrete value. It can be either Yes or No, 0 or 1, true or False, etc. but instead of giving the exact value as 0 and 1, it gives the probabilistic values which lie between 0 and 1.

● In Logistic regression, instead of fitting a regression line, we fit an "S" shaped logistic function, which predicts two maximum values (0 or 1).

● The curve from the logistic function indicates the likelihood of something such as whether the cells are cancerous or not, a mouse is obese or not based on its weight, etc.

● Logistic Regression is a significant machine learning algorithm because it has the ability to provide probabilities and classify new data using continuous and discrete datasets.

DECISION TREE ALGORITHM

Decision trees are a graphical representation of getting all the possible outcomes possible for a given decision-based query or problem. It is the closest algorithm to mimic human-like decisions. Also, the least amount of data cleaning or removing noisy data is required in this algorithm.

Algorithm:

- Step 1: Import the necessary dataset.
- Step 2: Create a train/test set. This division into subsets is necessary to predict the best possible values for the given attributes.
- Step 3: Build the necessary model for the required decision tree to make a prediction.
- Step 4: Make a prediction.
- Step 5: Measure the performance of the decision tree algorithm.

- Step 6: Tune the hyper-parameters accordingly. By doing this we are training the algorithm for fitting the model parameters.

RANDOM FOREST ALGORITHM

Random Forest (RF) is an ensemble ML algorithm based on multiple decision trees whose outcomes are merged, leading to gains in performance and stability while still being faster than other ensemble methods.

Algorithm RF works in two steps:

- First, for every tree, we obtain a sequence of instances, sampled randomly with replacement from the training set. Each sequence of instances corresponds to a random vector Θk characterizing a particular tree.

- As the sequences will be slightly different from one another, so will the decision trees constructed from them. The prediction of the kth tree for a given input x is given by where K is the number of trees.

- Each split of a tree uses a random selection of features to further avoid correlation. There are many ways to split a node S into two subsets. Assume a threshold c is chosen for the selected feature, splitting S into S1, S2 according to each feature value vi.

- For a regression task, one can use a c that minimizes the difference in the sum of squared errors

XGBOOST

Gradient boosting is a method standing out for its prediction speed and accuracy, particularly with large and complex datasets. From Kaggle competitions to machine learning solutions for business, this algorithm has produced the best results.

- We already know that errors play a major role in any machine learning algorithm. There are mainly two types of error, bias error and variance error.

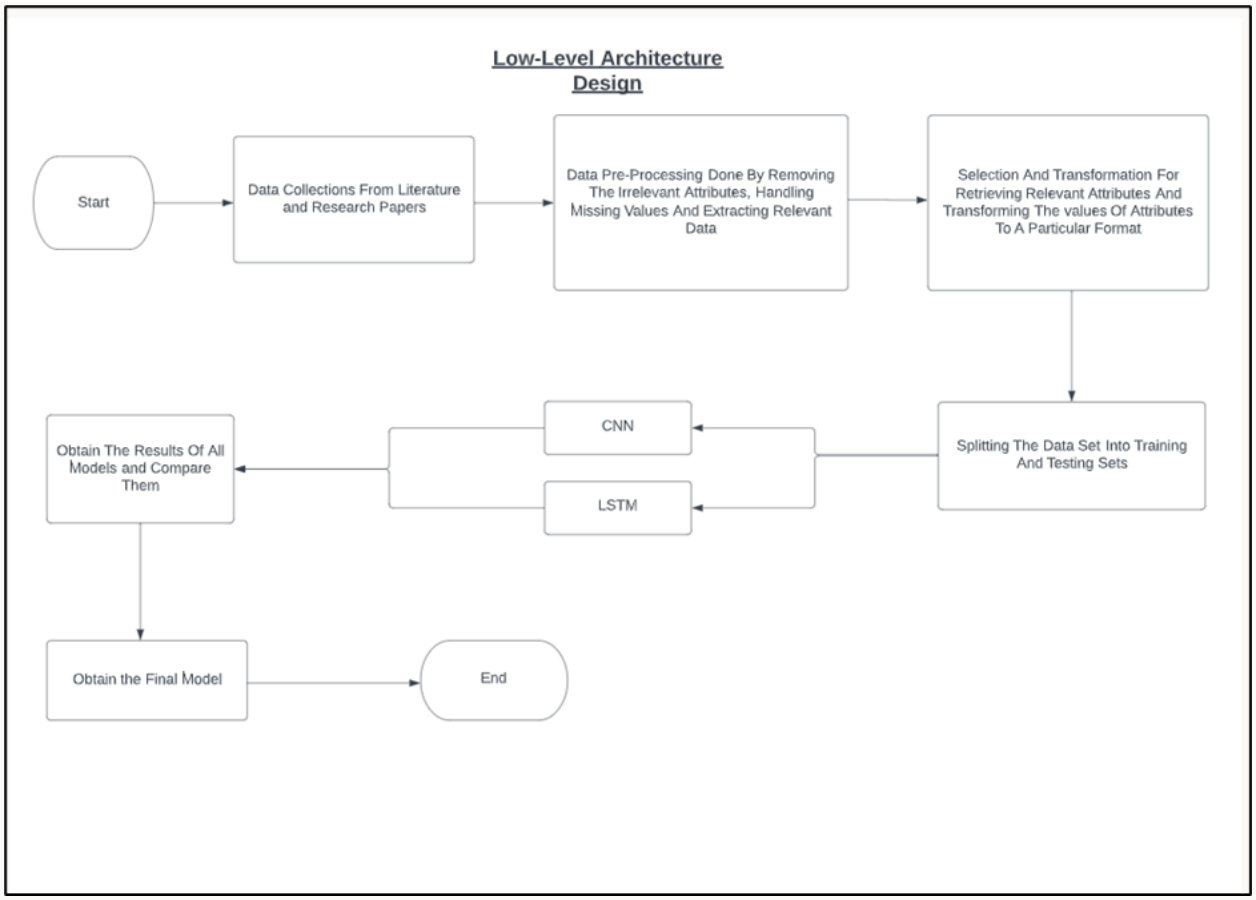
- Gradient boost algorithm helps us minimize bias error of the model. The main idea behind this algorithm is to build models sequentially and these subsequent models try to reduce the errors of the previous model.

- This is done by building a new model on the errors or residuals of the previous model. When the target column is continuous, we use Gradient Boosting Regressor whereas when it is a classification problem, we use Gradient Boosting Classifier.

- The only difference between the two is the “Loss function”. The objective here is to minimize this loss function by adding weak learners using gradient descent.

- Since it is based on loss function hence for regression problems, we’ll have different loss functions like Mean squared error (MSE) and for classification, we will have different for e.g. log-likelihood.

Low-Level Architecture Design for the implementation of Neural Networks:



Implementation of the Modules:

CNN

A Convolutional Neural Network (ConvNet/CNN) is a Deep Learning algorithm which can take in an input image, assign importance (learnable weights and biases) to various aspects/objects in the image and be able to differentiate one from the other. The pre-processing required in a ConvNet is much lower as compared to other classification algorithms.

- Convolution layers consist of a set of learnable filters (a patch in the above image). Every filter has a small width and height and the same depth as that of input volume (3 if the input layer is image input).
- For example, if we have to run convolution on an image with dimension 34x34x3. The possible size of filters can be axax3, where ‘a’ can be 3, 5, 7, etc but small as compared to image dimension.
- During forward pass, we slide each filter across the whole input volume step by step where each step is called stride (which can have value 2 or 3 or even 4 for high dimensional images) and compute the dot product between the weights of filters and patch from input volume.
- As we slide our filters we’ll get a 2-D output for each filter and we’ll stack them together and as a result, we’ll get output volume having a depth equal to the number of filters. The network will learn all the filters.

LSTM

LSTM is a special category of recurrent neural network aimed at handling time-series data. LSTM has a recurrently self-connected linear unit called “constant error carousel” (CEC). The recurrency of activation and error signals helps CEC provide short-term memory storage for the long term. In an LSTM unit, the memory part comprises a cell and there are three “regulators” called gates, controlling the passage of information inside the LSTM unit: an input gate, an output gate and a forget gate.

LSTM works in four steps:

1. Information to be forgotten is identified from previous time step using forget gate.
2. New information is sought for updating cell state using input gate and tanh.
3. Cell state is updated using the above two gates information.
4. Relevant information is yielded using output gate and the squashing function.

5. Results

LSTM

```
[ ]  def clean_str(string):
         """
         Cleaning of dataset
         """
         string = re.sub(r"\\", "", string)
         string = re.sub(r"\'", "", string)
         string = re.sub(r"\"", "", string)
         return string.strip().lower()
```

```
[ ]  data_train = pd.read_csv('data/train_Mixed.csv')
```

```
     # Input Data preprocessing
     data_train = pd.read_csv('data/train_Mixed.csv')
     #print(data_train.columns)
     #print('What the raw input data looks like:')
     #print(data_train[0:5])
     texts = []
     labels = []

     for i in range(data_train.text.shape[0]):
         text1 = data_train.title[i]
         text2 = data_train.text[i]
         text = str(text1) +""+ str(text2)
         texts.append(text)
         labels.append(data_train.label[i])

     tokenizer = Tokenizer(num_words=MAX_NB_WORDS)
     tokenizer.fit_on_texts(texts)
```

```
[ ] sequences = tokenizer.texts_to_sequences(texts)

    word_index = tokenizer.word_index
    #print('Found %s unique tokens.' % len(word_index))
```

```
[ ] # Pad input sequences
    data = pad_sequences(sequences, maxlen=MAX_SEQUENCE_LENGTH)
    labels = to_categorical(np.asarray(labels),num_classes = 2)
    print('Shape of data tensor:', data.shape)
    print('Shape of label tensor:', labels.shape)
```

```
Shape of data tensor: (20800, 1000)
Shape of label tensor: (20800, 2)
```

```
[ ] # Train test validation Split
    from sklearn.model_selection import train_test_split

    indices = np.arange(data.shape[0])
    np.random.shuffle(indices)
    data = data[indices]
    labels = labels[indices]
    x_train, x_test, y_train, y_test = train_test_split( data, labels, test_size=0.20, random_state=42)
    x_test, x_val, y_test, y_val = train_test_split( x_test, y_test, test_size=0.50, random_state=42)
    print('Size of train, validation, test:', len(y_train), len(y_val), len(y_test))

    print('real & fake news in train,valt,test:')
    print(y_train.sum(axis=0))
    print(y_val.sum(axis=0))
    print(y_test.sum(axis=0))
```

```
Size of train, validation, test: 16640 2080 2080
real & fake news in train,valt,test:
[8281. 8359.]
[1068. 1012.]
[1038. 1042.]
```

```
[ ]  _________________________________________________________________
     Layer (type)                 Output Shape              Param #
     =================================================================
     embedding_1 (Embedding)      (None, 1000, 100)         25187700
     _________________________________________________________________
     dropout_2 (Dropout)          (None, 1000, 100)         0
     _________________________________________________________________
     conv1d_3 (Conv1D)            (None, 1000, 32)          16032
     _________________________________________________________________
     max_pooling1d_3 (MaxPooling1 (None, 500, 32)           0
     _________________________________________________________________
     conv1d_4 (Conv1D)            (None, 500, 64)           6208
     _________________________________________________________________
     max_pooling1d_4 (MaxPooling1 (None, 250, 64)           0
     _________________________________________________________________
     lstm_2 (LSTM)                (None, 100)               66000
     _________________________________________________________________
     batch_normalization_2 (Batch (None, 100)               400
     _________________________________________________________________
     dense_5 (Dense)              (None, 256)               25856
     _________________________________________________________________
     dense_6 (Dense)              (None, 128)               32896
     _________________________________________________________________
     dense_7 (Dense)              (None, 64)                8256
     _________________________________________________________________
     dense_8 (Dense)              (None, 2)                 130
     =================================================================
     Total params: 25,343,478
     Trainable params: 25,343,278
     Non-trainable params: 200
     _________________________________________________________________
     None
     Epoch 1/10
     16640/16640 [==============================] - 250s 15ms/step - loss: 0.4041 - acc: 0.7993
     Epoch 2/10
     16640/16640 [==============================] - 247s 15ms/step - loss: 0.2620 - acc: 0.8854
     Epoch 3/10
```

```
Epoch 3/10
16640/16640 [==============================] - 248s 15ms/step - loss: 0.1345 - acc: 0.9460
Epoch 4/10
16640/16640 [==============================] - 248s 15ms/step - loss: 0.0653 - acc: 0.9754
Epoch 5/10
16640/16640 [==============================] - 248s 15ms/step - loss: 0.0338 - acc: 0.9881
Epoch 6/10
16640/16640 [==============================] - 248s 15ms/step - loss: 0.0189 - acc: 0.9931
Epoch 7/10
16640/16640 [==============================] - 252s 15ms/step - loss: 0.0142 - acc: 0.9945
Epoch 8/10
16640/16640 [==============================] - 249s 15ms/step - loss: 0.0077 - acc: 0.9974
Epoch 9/10
16640/16640 [==============================] - 250s 15ms/step - loss: 0.0091 - acc: 0.9968
Epoch 10/10
16640/16640 [==============================] - 255s 15ms/step - loss: 0.0105 - acc: 0.9974
```

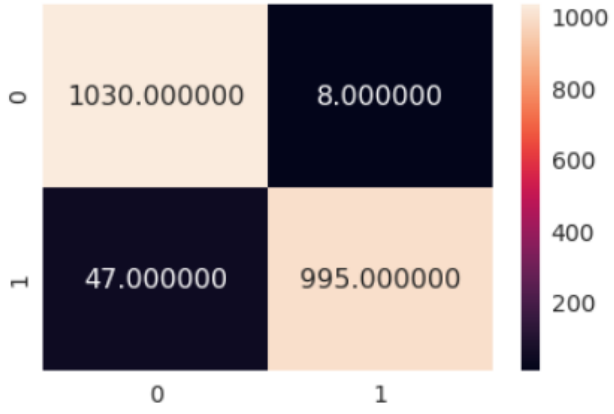
```
[ ] embedding_vecor_length = 32
    modelg = Sequential()
    modelg.add(embedding_layer)
    modelg.add(GRU(100, dropout=0.2, recurrent_dropout=0.2))
    ###
    modelg.add(BatchNormalization())
    ###
    modelg.add(Dense(2, activation='softmax'))
    modelg.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])
    print(modelg.summary())
    modelg.fit(x_train, y_train, epochs=2, batch_size=64)
    modelg.save('gru.h5')
```

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
embedding_1 (Embedding)      (None, 1000, 100)         25187700
_________________________________________________________________
```

```
[ ] _________________________________________________________________
    gru_1 (GRU)                  (None, 100)               60300
    _________________________________________________________________
    batch_normalization_3 (Batch (None, 100)               400
    _________________________________________________________________
    dense_3 (Dense)              (None, 2)                 202
    =================================================================
    Total params: 25,248,602
    Trainable params: 25,248,402
    Non-trainable params: 200
    _________________________________________________________________
    None
    Epoch 1/2
    16640/16640 [==============================] - 508s 31ms/step - loss: 0.3197 - acc: 0.8590
    Epoch 2/2
    16640/16640 [==============================] - 507s 30ms/step - loss: 0.1205 - acc: 0.9547
```

```
[ ]  # Test model 1
     import matplotlib.pyplot as plt
     import seaborn as sns
     test_preds = modell.predict(x_test)
     test_preds = np.round(test_preds)
     correct_predictions = float(sum(test_preds == y_test)[0])
     print("Correct predictions:", correct_predictions)
     print("Total number of test examples:", len(y_test))
     print("Accuracy of model1: ", correct_predictions/float(len(y_test)))

     # Creating the Confusion Matrix
     from sklearn.metrics import confusion_matrix
     x_pred = modell.predict(x_test)
     x_pred = np.round(x_pred)
     x_pred = x_pred.argmax(1)
     y_test_s = y_test.argmax(1)
     cm = confusion_matrix(y_test_s, x_pred)
     # plt.matshow(cm, cmap=plt.cm.binary, interpolation='nearest')
     # plt.title('Confusion matrix - model1')
     # plt.colorbar()
     # plt.ylabel('expected label')
     # plt.xlabel('predicted label')
     # plt.show()
     sns.set(font_scale=1.4)#for label size
     sns.heatmap(cm,annot=True,annot_kws={"size": 16},fmt='1f')# font size
```

```
Correct predictions: 2025.0
Total number of test examples: 2080
Accuracy of model1:  0.9735576923076923
<matplotlib.axes._subplots.AxesSubplot at 0x7f4c6c059588>
```



CNN:

```
[ ] data_train = pd.read_csv('data/train_Mixed.csv')
    data_train.text[1]

    'Ever get the feeling your life circles the roundabout rather than heads in a straight line toward the intended destination? [Hillary Clinton remains the big woman on campus in leafy, liberal Wellesley, Mas
```

```
[ ] # Input Data preprocessing
    data_train = pd.read_csv('data/train_Mixed.csv')
    #data_train['label'] = data_train['label'].replace('FAKE',1)
    #data_train['label'] = data_train['label'].replace('REAL',0)
    print(data_train.columns)
    print('What the raw input data looks like:')
    print(data_train[0:5])
    texts = []
    labels = []

    for i in range(data_train.text.shape[0]):
        text1 = data_train.title[i]
        text2 = data_train.text[i]
        text = str(text1) +""+ str(text2)
        texts.append(text)
        labels.append(data_train.label[i])

    tokenizer = Tokenizer(num_words=MAX_NB_WORDS)
    tokenizer.fit_on_texts(texts)
    sequences = tokenizer.texts_to_sequences(texts)

    word_index = tokenizer.word_index
    print('Found %s unique tokens.' % len(word_index))
```

```
Index(['id', 'title', 'author', 'text', 'label'], dtype='object')
What the raw input data looks like:
   id                                              title              author  \
0   0  House Dem Aide: We Didn’t Even See Comey’s Let...       Darrell Lucus
1   1  FLYNN: Hillary Clinton, Big Woman on Campus - ...     Daniel J. Flynn
2   2                  Why the Truth Might Get You Fired  Consortiumnews.com
3   3  15 Civilians Killed In Single US Airstrike Hav...     Jessica Purkiss
4   4  Iranian woman jailed for fictional unpublished...      Howard Portnoy

                                                text  label
0  House Dem Aide: We Didn’t Even See Comey’s Let...      1
1  Ever get the feeling your life circles the rou...      0
2  Why the Truth Might Get You Fired October 29, ...      1
3  Videos 15 Civilians Killed In Single US Airstr...      1
4  Print \nAn Iranian woman has been sentenced to...      1
Found 251876 unique tokens.
```

```
[ ] # Pad input sequences
    data = pad_sequences(sequences, maxlen=MAX_SEQUENCE_LENGTH)
    labels = to_categorical(np.asarray(labels),num_classes = 2)
    print('Shape of data tensor:', data.shape)
    print('Shape of label tensor:', labels.shape)
```

```
Shape of data tensor: (20800, 1000)
Shape of label tensor: (20800, 2)
```

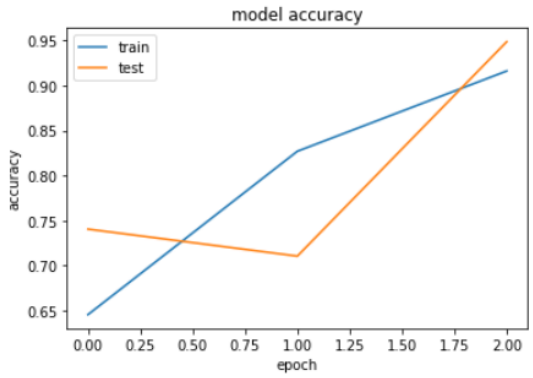
```
[ ]  # Train test validation Split
     from sklearn.model_selection import train_test_split

     indices = np.arange(data.shape[0])
     np.random.shuffle(indices)
     data = data[indices]
     labels = labels[indices]
     x_train, x_test, y_train, y_test = train_test_split( data, labels, test_size=0.20, random_state=42)
     x_test, x_val, y_test, y_val = train_test_split( data, labels, test_size=0.50, random_state=42)
     print('Size of train, validation, test:', len(y_train), len(y_val), len(y_test))

     print('real & fake news in train,valt,test:')
     print(y_train.sum(axis=0))
     print(y_val.sum(axis=0))
     print(y_test.sum(axis=0))
```

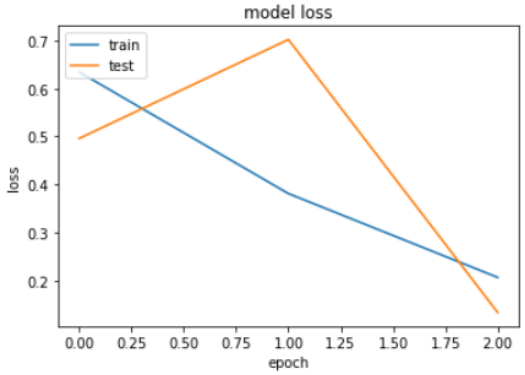
```
Size of train, validation, test: 16640 10400 10400
real & fake news in train,valt,test:
[8291. 8349.]
[5201. 5199.]
[5186. 5214.]
```

```
[ ]  # Simple CNN model
     sequence_input = Input(shape=(MAX_SEQUENCE_LENGTH,), dtype='int32')
     embedded_sequences = embedding_layer(sequence_input)
     l_cov1= Conv1D(128, 5, activation='relu')(embedded_sequences)
     l_pool1 = MaxPooling1D(5)(l_cov1)
     l_cov2 = Conv1D(128, 5, activation='relu')(l_pool1)
     l_pool2 = MaxPooling1D(5)(l_cov2)
     l_cov3 = Conv1D(128, 5, activation='relu')(l_pool2)
     l_pool3 = MaxPooling1D(35)(l_cov3)  # global max pooling
     l_flat = Flatten()(l_pool3)
     l_dense = Dense(128, activation='relu')(l_flat)
     preds = Dense(2, activation='softmax')(l_dense)

     model = Model(sequence_input, preds)
     model.compile(loss='categorical_crossentropy',
                   optimizer='adadelta',
                   metrics=['acc'])

     print("Fitting the simple convolutional neural network model")
     model.summary()
     history = model.fit(x_train, y_train, validation_data=(x_val, y_val),
               epochs=3, batch_size=128)
```

```
Fitting the simple convolutional neural network model
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_2 (InputLayer)         (None, 1000)              0
_________________________________________________________________
embedding_1 (Embedding)      (None, 1000, 100)         25187700
_________________________________________________________________
conv1d_6 (Conv1D)            (None, 996, 128)          64128
_________________________________________________________________
max_pooling1d_6 (MaxPooling1 (None, 199, 128)          0
_________________________________________________________________
conv1d_7 (Conv1D)            (None, 195, 128)          82048
_________________________________________________________________
max_pooling1d_7 (MaxPooling1 (None, 39, 128)           0
_________________________________________________________________
conv1d_8 (Conv1D)            (None, 35, 128)           82048
_________________________________________________________________
max_pooling1d_8 (MaxPooling1 (None, 1, 128)            0
_________________________________________________________________
flatten_2 (Flatten)          (None, 128)               0
_________________________________________________________________
dense_3 (Dense)              (None, 128)               16512
_________________________________________________________________
dense_4 (Dense)              (None, 2)                 258
=================================================================
Total params: 25,432,694
Trainable params: 25,432,694
Non-trainable params: 0
_________________________________________________________________
Train on 16640 samples, validate on 10400 samples
Epoch 1/3
16640/16640 [==============================] - 110s 7ms/step - loss: 0.5985 - acc: 0.6799 - val_loss: 0.4205 - val_acc: 0.7812
Epoch 2/3
16640/16640 [==============================] - 109s 7ms/step - loss: 0.2717 - acc: 0.8867 - val_loss: 0.2260 - val_acc: 0.9074
Epoch 3/3
16640/16640 [==============================] - 103s 6ms/step - loss: 0.1721 - acc: 0.9306 - val_loss: 0.2066 - val_acc: 0.9116
```

```
dict_keys(['val_loss', 'val_acc', 'loss', 'acc'])
```

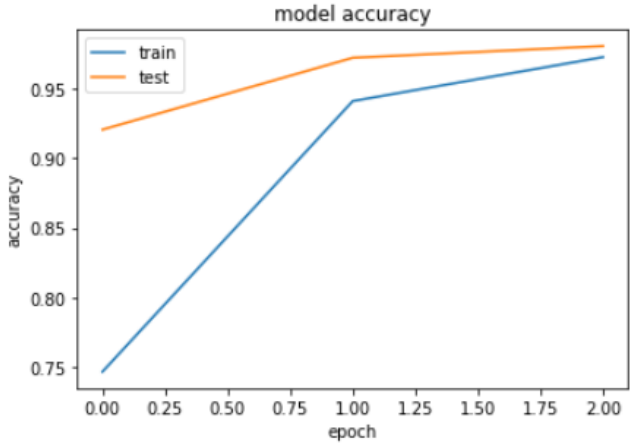


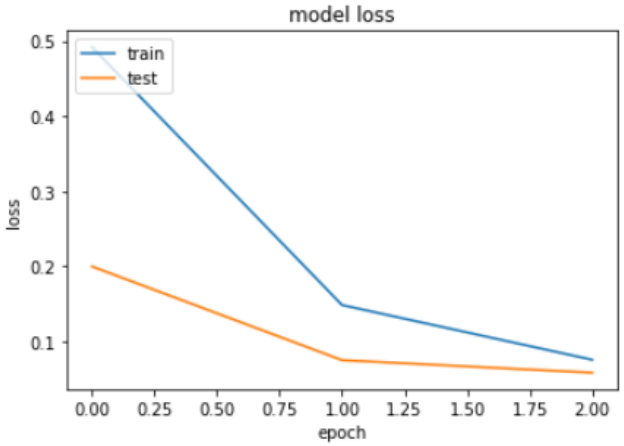


```
[ ]  #convolutional approach
     convs = []
     filter_sizes = [3,4,5]

     sequence_input = Input(shape=(MAX_SEQUENCE_LENGTH,), dtype='int32')
     embedded_sequences = embedding_layer(sequence_input)

     for fsz in filter_sizes:
         l_conv = Conv1D(nb_filter=128,filter_length=fsz,activation='relu')(embedded_sequences)
         l_pool = MaxPooling1D(5)(l_conv)
         convs.append(l_pool)

     l_merge = Merge(mode='concat', concat_axis=1)(convs)
     l_cov1= Conv1D(filters=128, kernel_size=5, activation='relu')(l_merge)
     l_pool1 = MaxPooling1D(5)(l_cov1)
     l_cov2 = Conv1D(filters=128, kernel_size=5, activation='relu')(l_pool1)
     l_pool2 = MaxPooling1D(30)(l_cov2)
     l_flat = Flatten()(l_pool2)
     l_dense = Dense(128, activation='relu')(l_flat)
     preds = Dense(2, activation='softmax')(l_dense)

     model2 = Model(sequence_input, preds)
     model2.compile(loss='categorical_crossentropy',
                    optimizer='adadelta',
                    metrics=['acc'])

     print("Fitting a more complex convolutional neural network model")
     model2.summary()
     history2 = model2.fit(x_train, y_train, validation_data=(x_val, y_val),
               epochs=3, batch_size=50)
     model2.save('model.h5')
```

```
/home/vin/anaconda3/lib/python3.6/site-packages/ipykernel_launcher.py:9: UserWarning: Update your `Conv1D` call to the Keras 2 API: `Conv1D(activation="relu", filters=128, kernel_size=3)`
  if __name__ == '__main__':
WARNING:tensorflow:From /home/vin/anaconda3/lib/python3.6/site-packages/tensorflow/python/util/deprecation.py:497: calling conv1d (from tensorflow.python.ops.nn_ops) with data_format=NHWC is deprecated a
Instructions for updating:
`NHWC` for data_format is deprecated, use `NWC` instead
Fitting a more complex convolutional neural network model
_________________________________________________________________________________________________
Layer (type)                 Output Shape              Param #     Connected to
=================================================================================================
input_1 (InputLayer)         (None, 1000)              0
_________________________________________________________________________________________________
embedding_1 (Embedding)      (None, 1000, 100)         25187700    input_1[0][0]
_________________________________________________________________________________________________
conv1d_1 (Conv1D)            (None, 998, 128)          38528       embedding_1[0][0]
_________________________________________________________________________________________________
conv1d_2 (Conv1D)            (None, 997, 128)          51328       embedding_1[0][0]
_________________________________________________________________________________________________
conv1d_3 (Conv1D)            (None, 996, 128)          64128       embedding_1[0][0]
_________________________________________________________________________________________________
max_pooling1d_1 (MaxPooling1D)  (None, 199, 128)       0           conv1d_1[0][0]
_________________________________________________________________________________________________
max_pooling1d_2 (MaxPooling1D)  (None, 199, 128)       0           conv1d_2[0][0]
_________________________________________________________________________________________________
max_pooling1d_3 (MaxPooling1D)  (None, 199, 128)       0           conv1d_3[0][0]
_________________________________________________________________________________________________
merge_1 (Merge)              (None, 597, 128)          0           max_pooling1d_1[0][0]
                                                                   max_pooling1d_2[0][0]
                                                                   max_pooling1d_3[0][0]
_________________________________________________________________________________________________
conv1d_4 (Conv1D)            (None, 593, 128)          82048       merge_1[0][0]
_________________________________________________________________________________________________
max_pooling1d_4 (MaxPooling1D)  (None, 118, 128)       0           conv1d_4[0][0]
_________________________________________________________________________________________________
conv1d_5 (Conv1D)            (None, 114, 128)          82048       max_pooling1d_4[0][0]
_________________________________________________________________________________________________
max_pooling1d_5 (MaxPooling1D)  (None, 3, 128)         0           conv1d_5[0][0]
_________________________________________________________________________________________________
flatten_1 (Flatten)          (None, 384)               0           max_pooling1d_5[0][0]
_________________________________________________________________________________________________
dense_1 (Dense)              (None, 128)               49280       flatten_1[0][0]
_________________________________________________________________________________________________
dense_2 (Dense)              (None, 2)                 258         dense_1[0][0]
=================================================================================================
Total params: 25,555,318
Trainable params: 25,555,318
Non-trainable params: 0
_________________________________________________________________________________________________
/home/vin/anaconda3/lib/python3.6/site-packages/ipykernel_launcher.py:9: UserWarning: Update your `Conv1D` call to the Keras 2 API: `Conv1D(activation="relu", filters=128, kernel_size=4)`
  if __name__ == '__main__':
/home/vin/anaconda3/lib/python3.6/site-packages/ipykernel_launcher.py:9: UserWarning: Update your `Conv1D` call to the Keras 2 API: `Conv1D(activation="relu", filters=128, kernel_size=5)`
  if __name__ == '__main__':
/home/vin/anaconda3/lib/python3.6/site-packages/ipykernel_launcher.py:13: UserWarning: The `Merge` layer is deprecated and will be removed after 08/2017. Use instead layers from `keras.layers.merge`, e.g
  del sys.path[0]
Train on 16640 samples, validate on 10400 samples
Epoch 1/3
16640/16640 [==============================] - 311s 19ms/step - loss: 0.4919 - acc: 0.7470 - val_loss: 0.1996 - val_acc: 0.9204
Epoch 2/3
```

```
dict_keys(['val_loss', 'val_acc', 'loss', 'acc'])
```



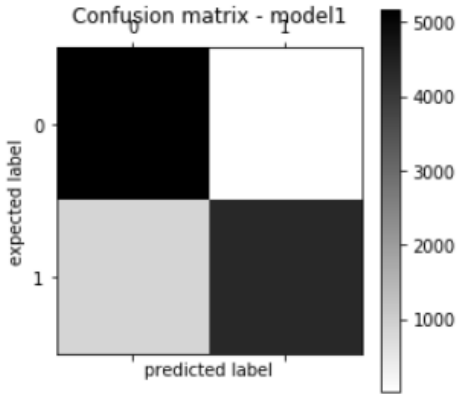


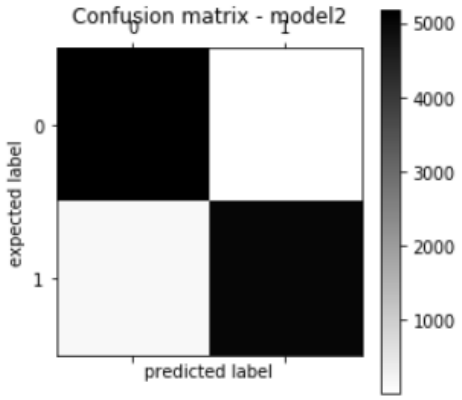
```
[ ]  # Test model 1
     test_preds = model.predict(x_test)
     test_preds = np.round(test_preds)
     correct_predictions = float(sum(test_preds == y_test)[0])
     print("Correct predictions:", correct_predictions)
     print("Total number of test examples:", len(y_test))
     print("Accuracy of model1: ", correct_predictions/float(len(y_test)))

     # Creating the Confusion Matrix
     from sklearn.metrics import confusion_matrix
     x_pred = model.predict(x_test)
     x_pred = np.round(x_pred)
     x_pred = x_pred.argmax(1)
     y_test_s = y_test.argmax(1)
     cm = confusion_matrix(y_test_s, x_pred)
     plt.matshow(cm, cmap=plt.cm.binary, interpolation='nearest')
     plt.title('Confusion matrix - model1')
     plt.colorbar()
     plt.ylabel('expected label')
     plt.xlabel('predicted label')
     # plt.show()

     #Test model 2
     test_preds2 = model2.predict(x_test)
     test_preds2 = np.round(test_preds2)
     correct_predictions = float(sum(test_preds2 == y_test)[0])
     print("Correct predictions:", correct_predictions)
     print("Total number of test examples:", len(y_test))
     print("Accuracy of model2: ", correct_predictions/float(len(y_test)))
```

```
    # Creating the Confusion Matrix
    x_pred = model2.predict(x_test)
    x_pred = np.round(x_pred)
    x_pred = x_pred.argmax(1)
    y_test_s = y_test.argmax(1)
    cm = confusion_matrix(y_test_s, x_pred)
    plt.matshow(cm, cmap=plt.cm.binary, interpolation='nearest',)
    plt.title('Confusion matrix - model2')
    plt.colorbar()
    plt.ylabel('expected label')
    plt.xlabel('predicted label')
    plt.show()
```

[ ]

```
Correct predictions: 9516.0
Total number of test examples: 10400
Accuracy of model1:  0.915
Correct predictions: 10229.0
Total number of test examples: 10400
Accuracy of model2:  0.9835576923076923
```





[ ]

```
score = model.evaluate(x_test, y_test, verbose=0)
print('Test loss:', score[0])
print('Test accuracy:', score[1])
```

```
Test loss: 0.19623974713568504
Test accuracy: 0.915
```

[ ]

```
score = model2.evaluate(x_test, y_test, verbose=0)
print('Test loss:', score[0])
print('Test accuracy:', score[1])
```

```
Test loss: 0.04651976277550252
Test accuracy: 0.9835576923076923
```

Reference No./Algorithm Name	Accuracy
[1]	0.9132
[4]*	0.9400
[15]	0.9186
[14]	0.90
[8]	0.9836
[5]	0.8649
CNN Model 1**	0.915
CNN Model 2**	0.98355
LSTM**	0.97355

*The given values are of the best observed algorithm, amongst many that were compared.
**The given deep learning algorithms are those that were implemented by us as part of Review 3

The above table draws a comparison between the top 6 observed accuracies from the literature review and the Neural networks (CNN & LSTM) implemented by us over the dataset. Here we make an observation that our implementation of CNN model 2 gave the highest competitive accuracy of 0.98355 closest to the accuracy observed in [8] of 0.9836.

6. Conclusion

In this project, we have studied the fake news article, creator and subject detection problem, i.e., the fake news detection problem. Based on the news augmented heterogeneous social network, a set of explicit and latent features can be extracted from the textual information of news articles, creators and subjects respectively. Furthermore, based on the connections among news articles, creators and news subjects, multiple models have been proposed for incorporating the network structure information into model learning. In this project the likes of XGBoost, Random Forest, Logistic Regression, Decision Tree, CNN and LSTM have been explored for the purpose of fake news detection. From the above it was observed that CNN provided the highest competitive accuracy of 0.98355 even when compared to the various models proposed in the literature papers that were surveyed.

7. References

Google Drive link to the literature papers:
https://drive.google.com/drive/folders/1RA-BhAosi-WnXHY8cJ5-HMrUT0LLt_xD?usp=sharing

[Base Research Paper]
Zhang, J., Dong, B., & Philip, S. Y. (2020, April). Fakedetector: Effective fake news detection with deep diffusive neural network. In 2020 IEEE 36th International Conference on Data Engineering (ICDE) (pp. 1826-1829). IEEE.

[19BIT0176]
[1] Deepak, S., & Chitturi, B. (2020). Deep neural approach to Fake-News identification. Procedia Computer Science, 167, 2236-2243..
[2] Chauhan, T., & Palivela, H. (2021). Optimization and improvement of fake news detection using deep learning approaches for societal benefit. International Journal of Information Management Data Insights, 1(2), 100051.
[3] Saikh, T., Anand, A., Ekbal, A., & Bhattacharyya, P. (2019, June). A novel approach towards fake news detection: deep learning augmented with textual entailment features. In International Conference on Applications of Natural Language to Information Systems (pp. 345-358). Springer, Cham.
[4] Ahmad, I., Yousaf, M., Yousaf, S., & Ahmad, M. O. (2020). Fake news detection using machine learning ensemble methods. Complexity, 2020.

[19BIT0206]
[5] Kaliyar, R. K., Goswami, A., & Narang, P. (2021). DeepFakE: improving fake news detection using tensor decomposition-based deep neural network. The Journal of Supercomputing, 77(2), 1015-1037.
[6] Agarwal, A., Mittal, M., Pathak, A., & Goyal, L. M. (2020). Fake news detection using a blend of neural networks: An application of deep learning. SN Computer Science, 1(3), 1-9.

[7] Zhang, J., Cui, L., Fu, Y., & Gouza, F. B. (2018). Fake news detection with deep diffusive network model. arXiv preprint arXiv:1805.08751.
[8] Kaliyar, R. K., Goswami, A., Narang, P., & Sinha, S. (2020). FNDNet–a deep convolutional neural network for fake news detection. Cognitive Systems Research, 61, 32-44.

[19BIT0208]
[9] Jain, A., Shakya, A., Khatter, H., & Gupta, A. K. (2019, September). A smart system for fake news detection using machine learning. In 2019 International Conference on Issues and Challenges in Intelligent Computing Techniques (ICICT) (Vol. 1, pp. 1-4). IEEE.
[10] Rashkin, H., Choi, E., Jang, J. Y., Volkova, S., & Choi, Y. (2017, September). Truth of varying shades: Analyzing language in fake news and political fact-checking. In Proceedings of the 2017 conference on empirical methods in natural language processing (pp. 2931-2937).
[11] Tacchini, E., Ballarin, G., Della Vedova, M. L., Moret, S., & de Alfaro, L. (2017). Some like it hoax: Automated fake news detection in social networks. arXiv preprint arXiv:1704.07506.
[12] Ruchansky, N., Seo, S., & Liu, Y. (2017, November). Csi: A hybrid deep model for fake news detection. In Proceedings of the 2017 ACM on Conference on Information and Knowledge Management (pp. 797-806).

[19BIT0256]
[13] Bahad, P., Saxena, P., & Kamal, R. (2019). Fake news detection using bi-directional LSTM-recurrent neural network. Procedia Computer Science, 165, 74-82.
[14] Rodríguez, Á. I., & Iglesias, L. L. (2019). Fake news detection using Deep Learning. arXiv preprint arXiv:1910.03496.
[15] Yang, Y., Zheng, L., Zhang, J., Cui, Q., Li, Z., & Yu, P. S. (2018). TI-CNN: Convolutional neural networks for fake news detection. arXiv preprint arXiv:1806.00749.
[16] Mayank, M., Sharma, S., & Sharma, R. (2021). DEAP-FAKED: Knowledge Graph based Approach for Fake News Detection. arXiv preprint arXiv:2107.10648

8. Video Links:

https://drive.google.com/drive/folders/1uU_j9vOdY9HFpUYjlhwLGm2h0gkk2BSg?usp=sharing