CSPC54 AI-ML
Project Detailed Design

Group 14

Roll Numbers

106119112
106119100

Group Members

SATYARTH PANDEY
RAJNEESH PANDEY

Section

CSE-B

A Deep Learning Approach for Automatic Detection of Fake News

Algorithm

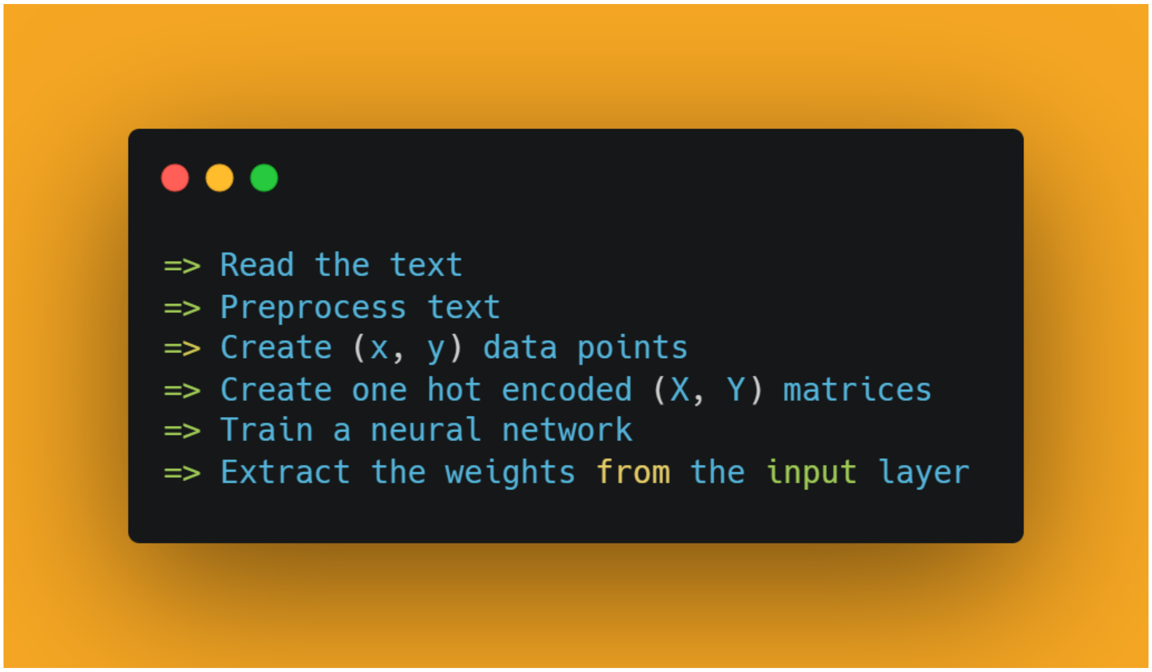
Model 1 :

Step-1: Pre-processing the Dataset in *.txt* format to split into Title, Content, and Label (in Training Dataset).

Step-2: Pass title and content for Word Embedding

Step-3 : Word Embedding :

Word embedding is a set of language modelling and feature learning techniques in natural language processing (NLP) where words or phrases from the vocabulary are mapped to vectors of real numbers.

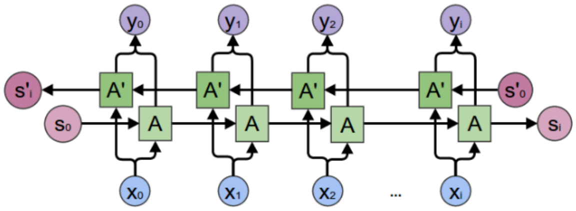


```
=> Read the text
=> Preprocess text
=> Create (x, y) data points
=> Create one hot encoded (X, Y) matrices
=> Train a neural network
=> Extract the weights from the input layer
```

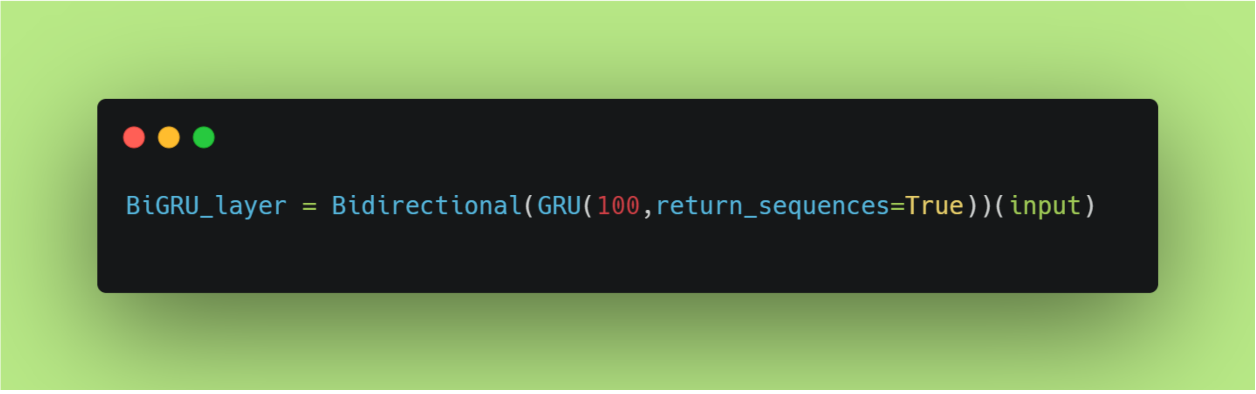
Step 4: After Word Embedding of inputs the resultant vectors passed to BiGRU Encoding Layer.

Step 5 : Bidirectional GRU's are a type of bidirectional recurrent neural networks with only the input and forget gates.

It allows for the use of information from both previous time steps and later time steps to make predictions about the current state.

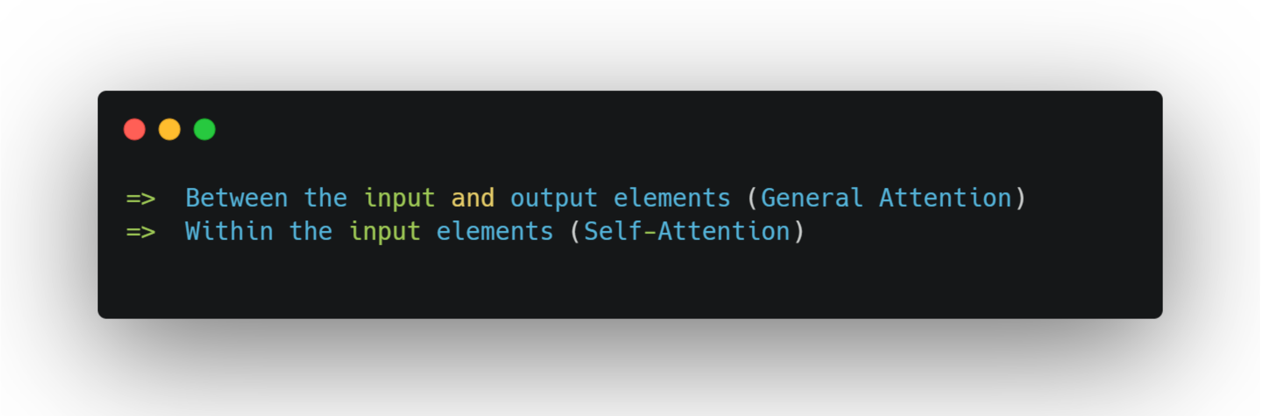


General Structure of Bidirectional Recurrent Neural Networks



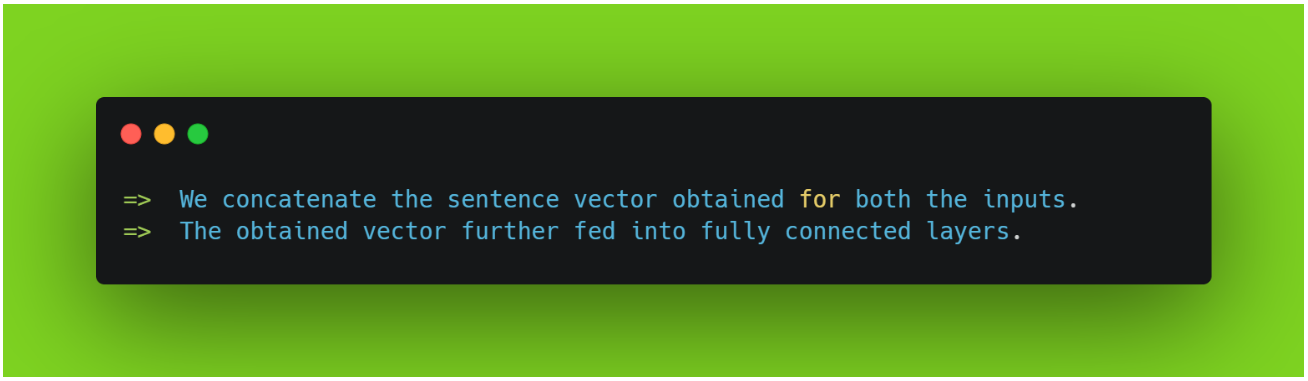
```
BiGRU_layer = Bidirectional(GRU(100,return_sequences=True))(input)
```

Step 6 : Attention is one **component** of a network's architecture, and is in charge of managing and quantifying the **interdependence**:



```
=>  Between the input and output elements (General Attention)
=>  Within the input elements (Self-Attention)
```

Step 7 : Merge Input Content and Input Title from above layers into Nodes of MLP (Multilayer Layer Perceptron):

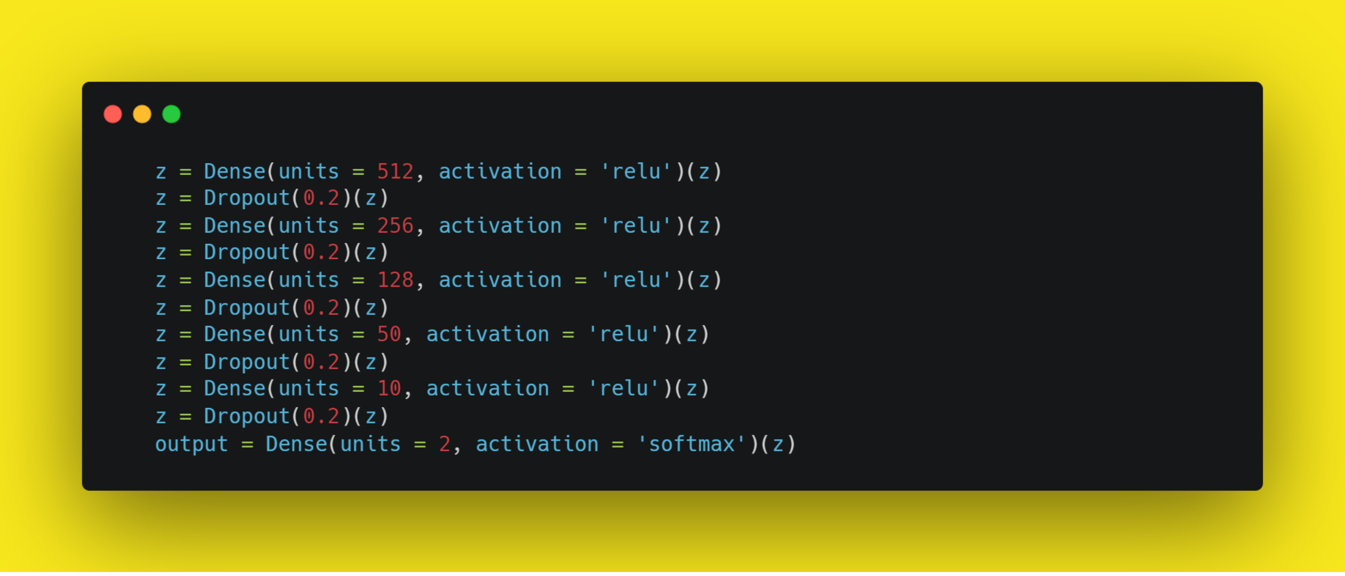


```
=>  We concatenate the sentence vector obtained for both the inputs.
=>  The obtained vector further fed into fully connected layers.
```

Step 8 : Five Layered MLP :

We use 512, 256, 128, 50 and 10 neurons, respectively, for five such layers with ReLU activation in each layer.

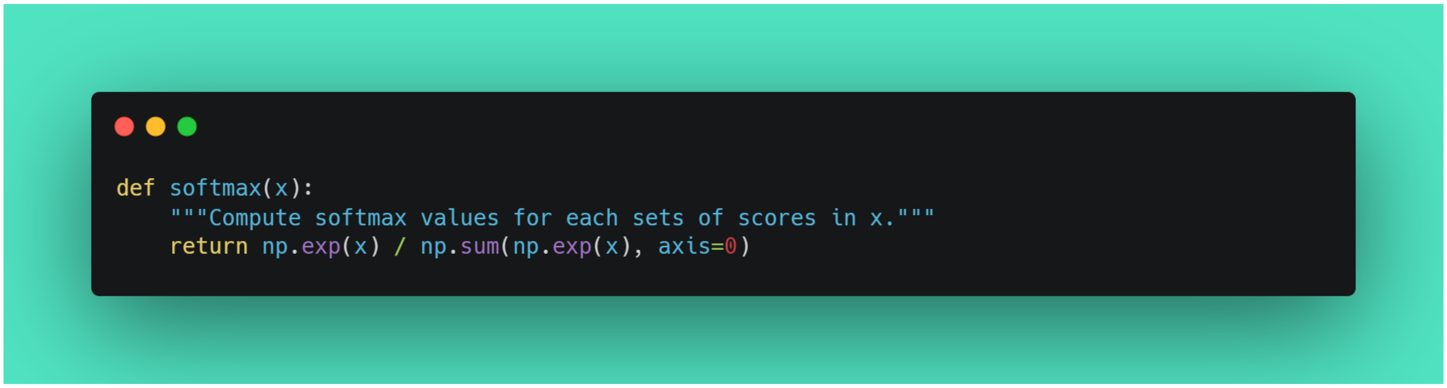
Between each such layer, we employ 20% dropout as a measurement of regularization.



```
    z = Dense(units = 512, activation = 'relu')(z)
    z = Dropout(0.2)(z)
    z = Dense(units = 256, activation = 'relu')(z)
    z = Dropout(0.2)(z)
    z = Dense(units = 128, activation = 'relu')(z)
    z = Dropout(0.2)(z)
    z = Dense(units = 50, activation = 'relu')(z)
    z = Dropout(0.2)(z)
    z = Dense(units = 10, activation = 'relu')(z)
    z = Dropout(0.2)(z)
    output = Dense(units = 2, activation = 'softmax')(z)
```

Step 9 : Two Way Softmax Layer

Finally, the output from the last fully connected layer is fed into a final classification layer with softmax activation function having 2 neurons.



```
def softmax(x):
    """Compute softmax values for each sets of scores in x."""
    return np.exp(x) / np.sum(np.exp(x), axis=0)
```

Step 10 :

We obtain final classification Output Label as Fake or Legit

Model 2 :

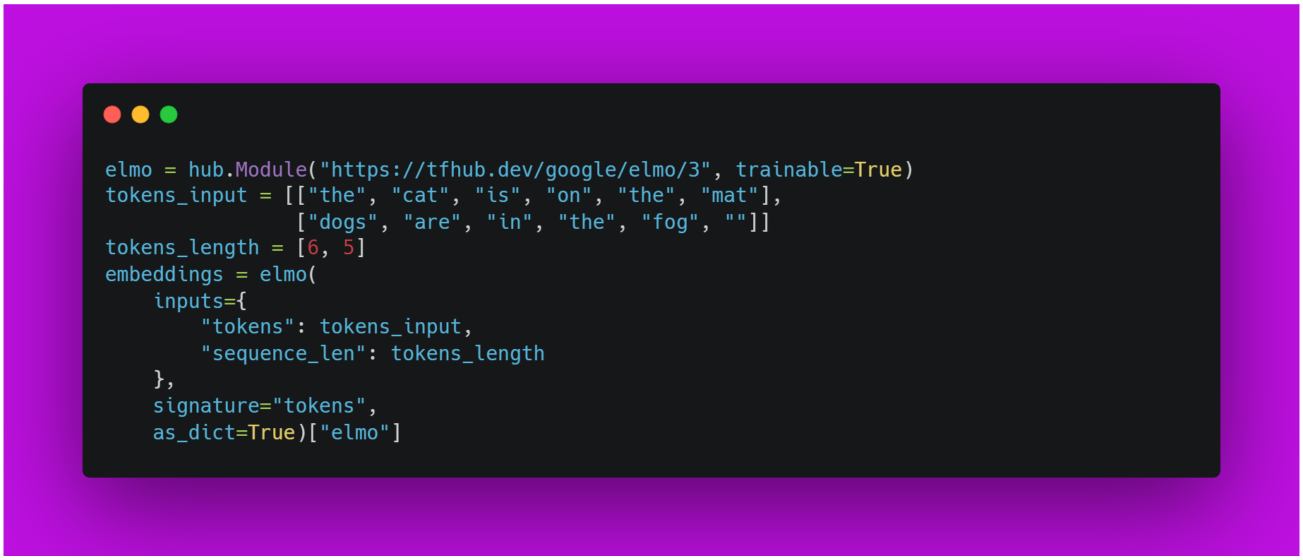
Step-1: Pre-processing the Dataset in *.txt* format to split into Title, Content, and Label (in Training Dataset).

Step 2 : Embedding Layer: Embedding from Language Model (ELMo) :

i) Contextual i.e. representation of each word is based on entire corpus in which it is used

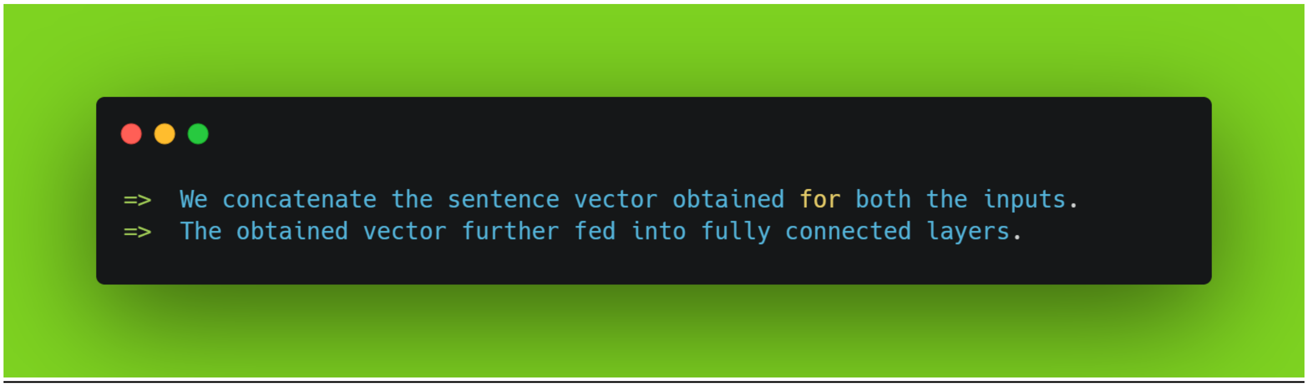
ii) Deep i.e. it combines all layers of a deep pre-trained neural network

iii) Character based i.e. it provides representations which are based on character, thus allowing the network to make use of morphological clues to form robust representation of out-of-vocabulary tokens during training



```
elmo = hub.Module("https://tfhub.dev/google/elmo/3", trainable=True)
tokens_input = [["the", "cat", "is", "on", "the", "mat"],
                ["dogs", "are", "in", "the", "fog", ""]]
tokens_length = [6, 5]
embeddings = elmo(
    inputs={
        "tokens": tokens_input,
        "sequence_len": tokens_length
    },
    signature="tokens",
    as_dict=True)["elmo"]
```

Step 3 : Merge Input Content and Input Title from above layers into Nodes of MLP (Multilayer Layer Perceptron):

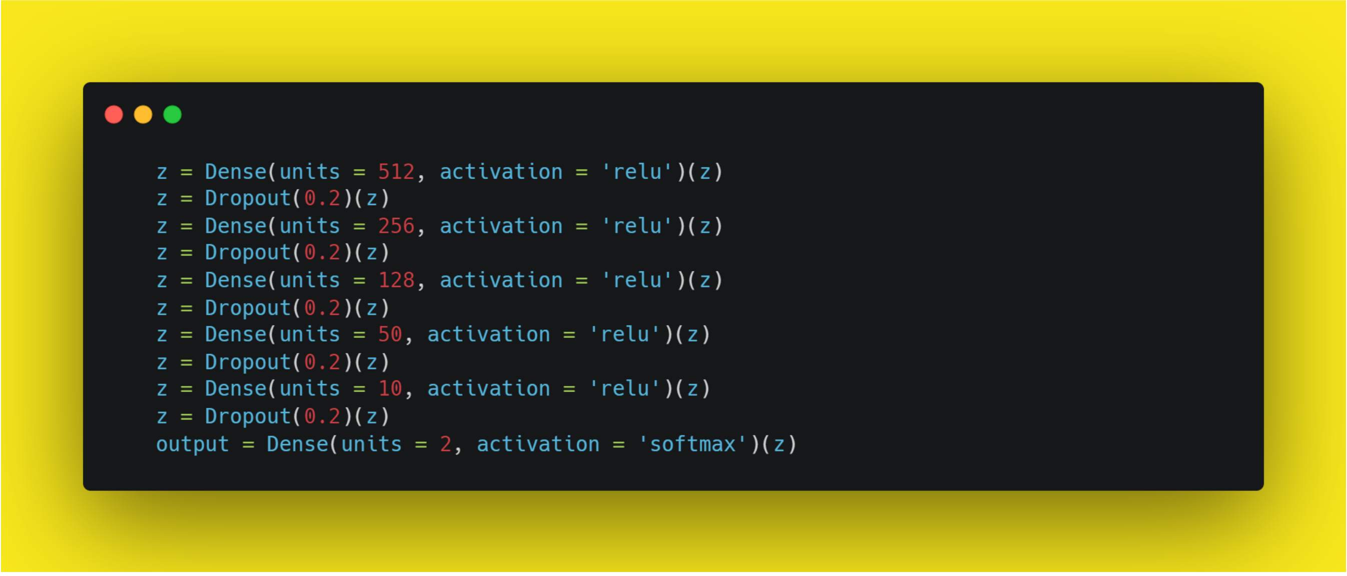


```
=>  We concatenate the sentence vector obtained for both the inputs.
=>  The obtained vector further fed into fully connected layers.
```

Step 4 : Five Layered MLP :

We use 512, 256, 128, 50 and 10 neurons, respectively, for five such layers with ReLU activation in each layer.

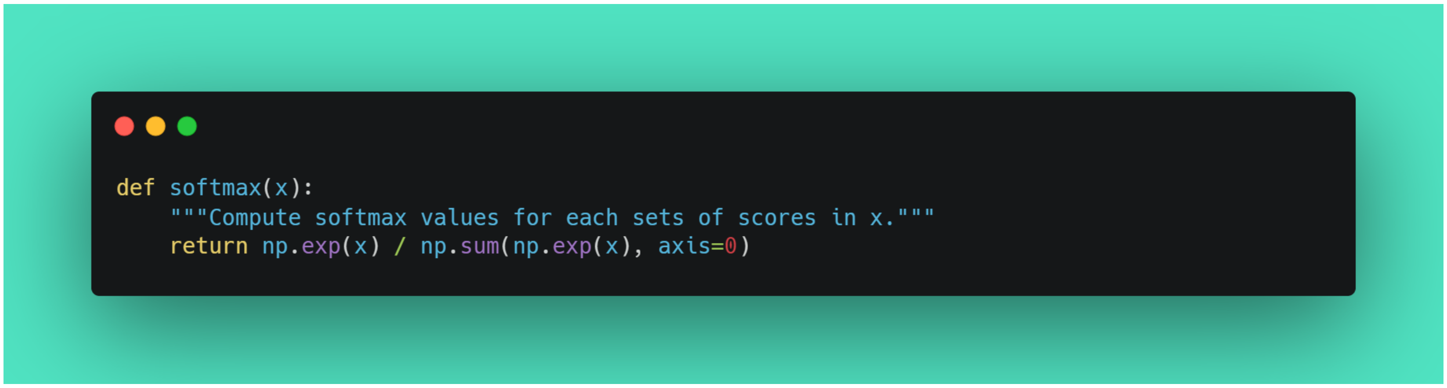
Between each such layer, we employ 20% dropout as a measurement of regularization.



```
    z = Dense(units = 512, activation = 'relu')(z)
    z = Dropout(0.2)(z)
    z = Dense(units = 256, activation = 'relu')(z)
    z = Dropout(0.2)(z)
    z = Dense(units = 128, activation = 'relu')(z)
    z = Dropout(0.2)(z)
    z = Dense(units = 50, activation = 'relu')(z)
    z = Dropout(0.2)(z)
    z = Dense(units = 10, activation = 'relu')(z)
    z = Dropout(0.2)(z)
    output = Dense(units = 2, activation = 'softmax')(z)
```

Step 5 : Two Way Softmax Layer

Finally, the output from the last fully connected layer is fed into a final classification layer with softmax activation function having 2 neurons.



```python
def softmax(x):
    """Compute softmax values for each sets of scores in x."""
    return np.exp(x) / np.sum(np.exp(x), axis=0)
```

Step 6 :

We obtain final classification Output Label as Fake or Legit.

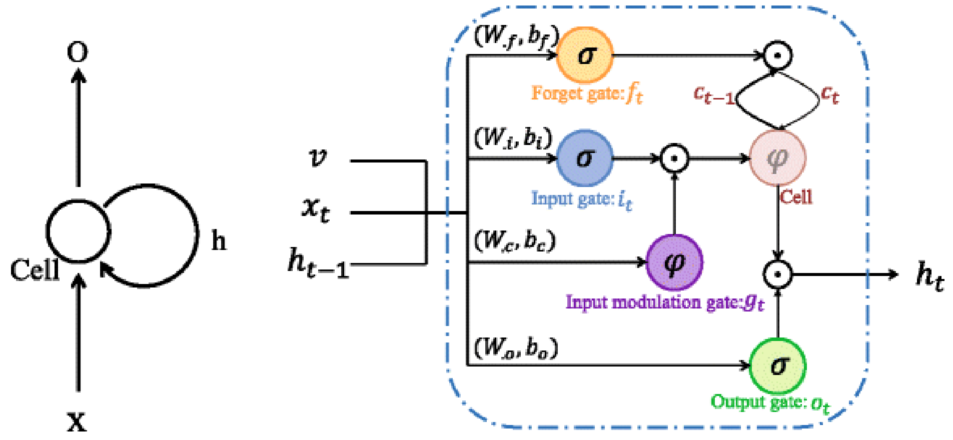
Modified Algorithm

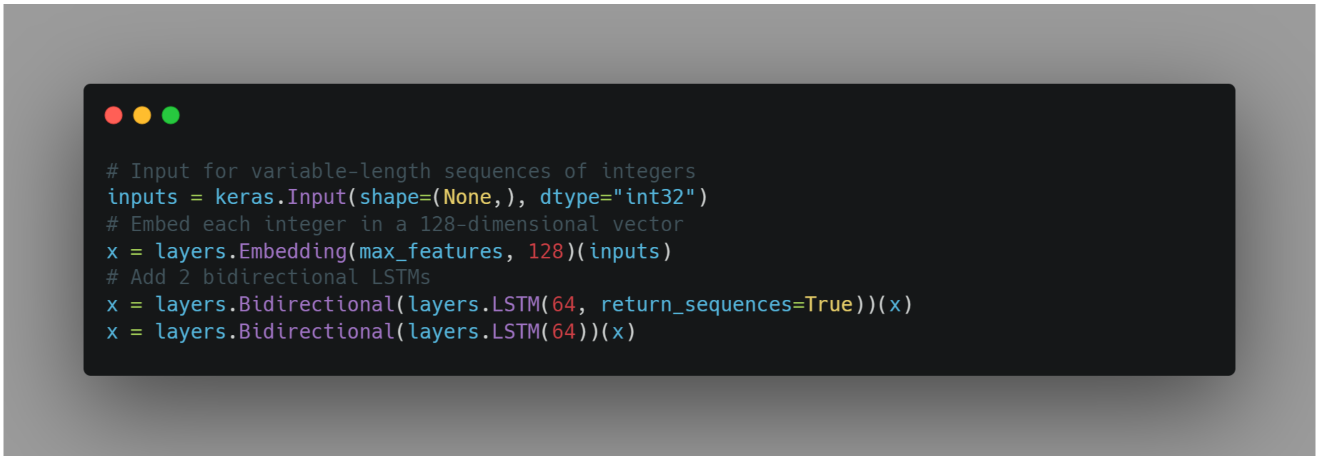
Step-1: Pre-processing the Dataset in *.txt* format to split into Title, Content, and Label (in Training Dataset).

Step-2 : LSTM Based architecture :

A Bidirectional LSTMs are an extension of traditional LSTMs that can improve model performance on sequence classification problems.

Obtaining model hyperparameters using Bayesian Optimization

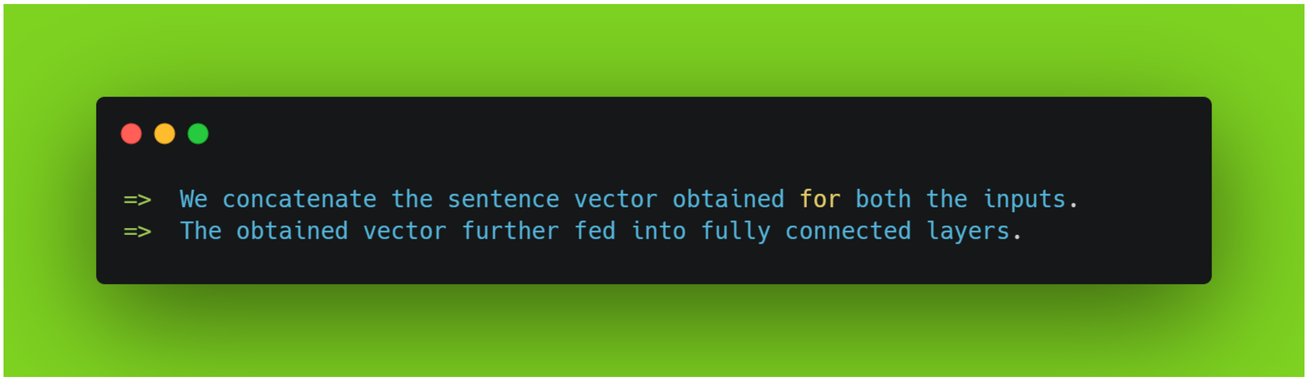




```
# Input for variable-length sequences of integers
inputs = keras.Input(shape=(None,), dtype="int32")
# Embed each integer in a 128-dimensional vector
x = layers.Embedding(max_features, 128)(inputs)
# Add 2 bidirectional LSTMs
x = layers.Bidirectional(layers.LSTM(64, return_sequences=True))(x)
x = layers.Bidirectional(layers.LSTM(64))(x)
```

Note: *Here onwards the continuation steps are same as Previous Algorithms.*

Step 3 : Merge Input Content and Input Title from above layers into Nodes of MLP (Multilayer Layer Perceptron):

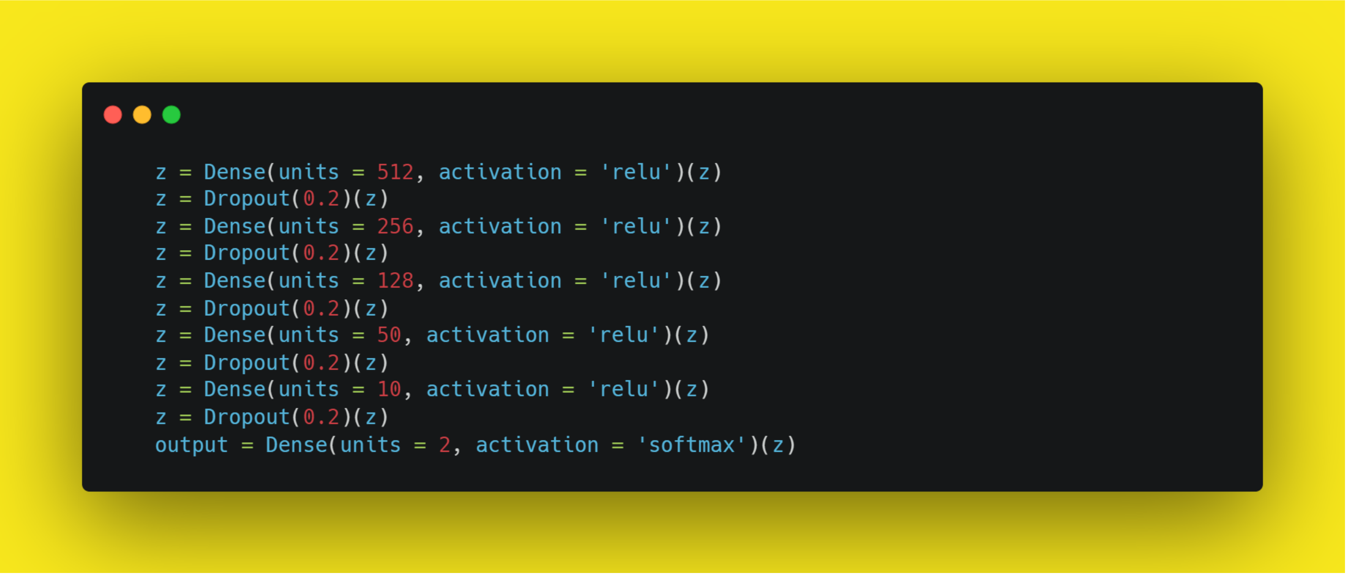


```
=>  We concatenate the sentence vector obtained for both the inputs.
=>  The obtained vector further fed into fully connected layers.
```

Step 4 *:* Five Layered MLP :

We use 512, 256, 128, 50 and 10 neurons, respectively, for five such layers with ReLU activation in each layer.

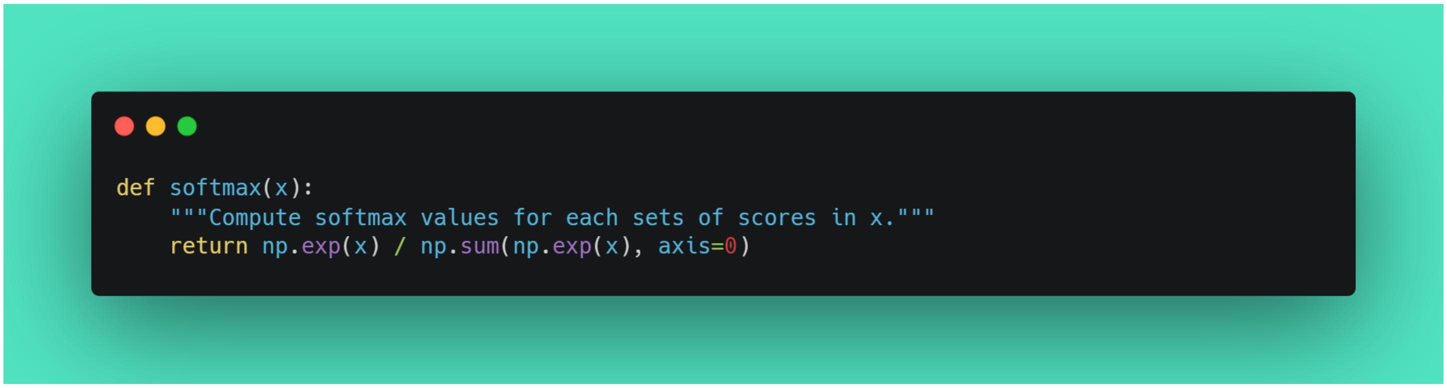
Between each such layer, we employ 20% dropout as a measurement of regularization.



```
z = Dense(units = 512, activation = 'relu')(z)
z = Dropout(0.2)(z)
z = Dense(units = 256, activation = 'relu')(z)
z = Dropout(0.2)(z)
z = Dense(units = 128, activation = 'relu')(z)
z = Dropout(0.2)(z)
z = Dense(units = 50, activation = 'relu')(z)
z = Dropout(0.2)(z)
z = Dense(units = 10, activation = 'relu')(z)
z = Dropout(0.2)(z)
output = Dense(units = 2, activation = 'softmax')(z)
```

Step 5 : Two Way Softmax Layer

Finally, the output from the last fully connected layer is fed into a final classification layer with softmax activation function having 2 neurons.

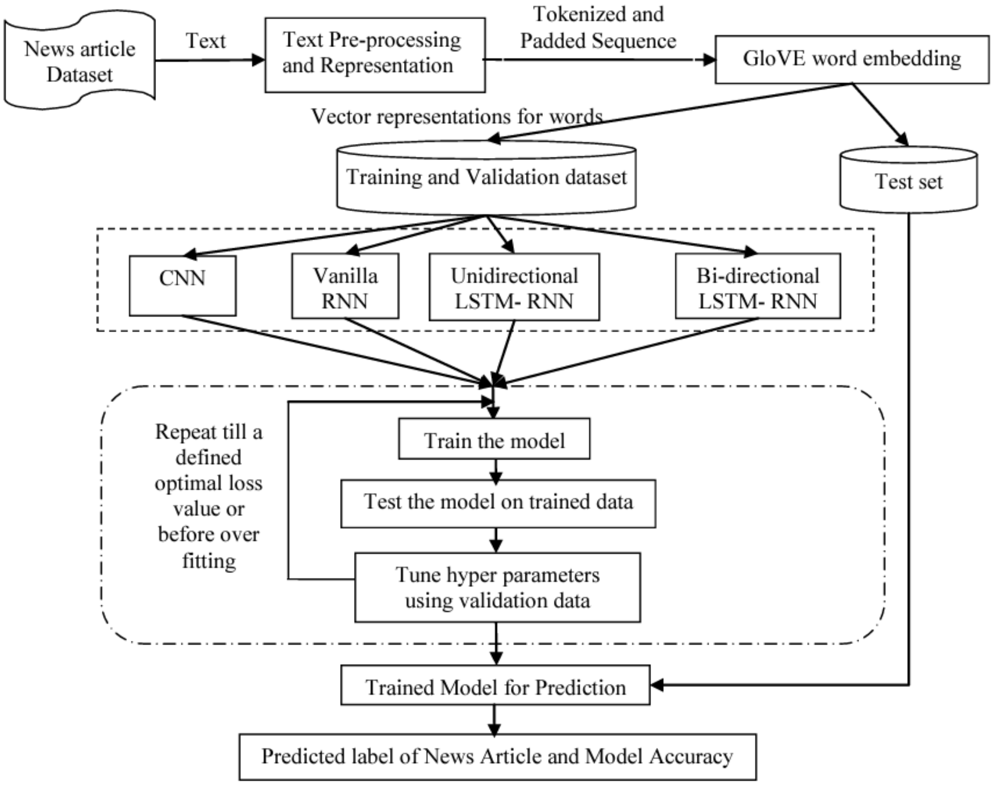


```python
def softmax(x):
    """Compute softmax values for each sets of scores in x."""
    return np.exp(x) / np.sum(np.exp(x), axis=0)
```

Step 6 :

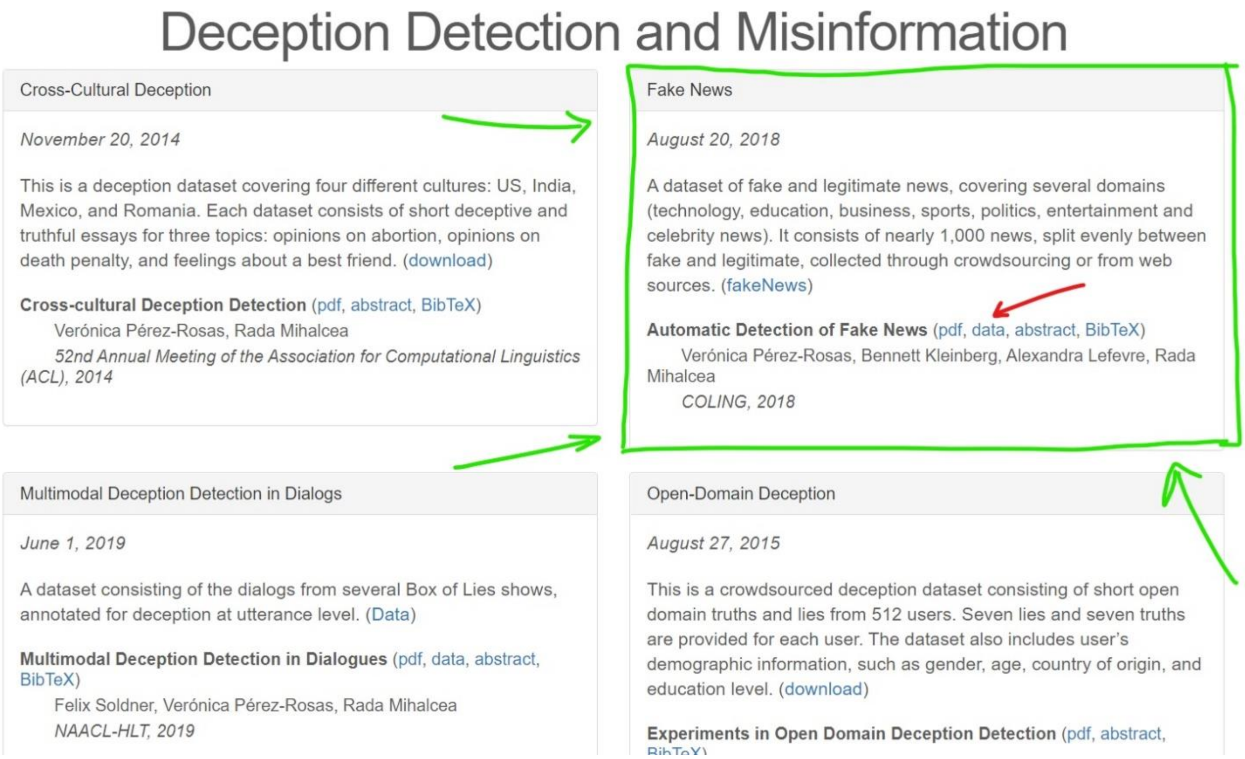
We obtain final classification Output Label as Fake or Legit

Flow Diagram of Modified Algorithm:


News article Dataset
Text
Text Pre-processing and Representation
Tokenized and Padded Sequence
GloVE word embedding
Vector representations for words
Training and Validation dataset
Test set
CNN
Vanilla RNN
Unidirectional LSTM- RNN
Bi-directional LSTM- RNN
Repeat till a defined optimal loss value or before over fitting
Train the model
Test the model on trained data
Tune hyper parameters using validation data
Trained Model for Prediction
Predicted label of News Article and Model Accuracy

Link to dataset

Official Dataset : https://lit.eecs.umich.edu/downloads.html#Fake%20News



Deception Detection and Misinformation

Cross-Cultural Deception

November 20, 2014

This is a deception dataset covering four different cultures: US, India, Mexico, and Romania. Each dataset consists of short deceptive and truthful essays for three topics: opinions on abortion, opinions on death penalty, and feelings about a best friend. (download)

Cross-cultural Deception Detection (pdf, abstract, BibTeX)
Verónica Pérez-Rosas, Rada Mihalcea
52nd Annual Meeting of the Association for Computational Linguistics (ACL), 2014

Fake News

August 20, 2018

A dataset of fake and legitimate news, covering several domains (technology, education, business, sports, politics, entertainment and celebrity news). It consists of nearly 1,000 news, split evenly between fake and legitimate, collected through crowdsourcing or from web sources. (fakeNews)

Automatic Detection of Fake News (pdf, data, abstract, BibTeX)
Verónica Pérez-Rosas, Bennett Kleinberg, Alexandra Lefevre, Rada Mihalcea
COLING, 2018

Multimodal Deception Detection in Dialogs

June 1, 2019

A dataset consisting of the dialogs from several Box of Lies shows, annotated for deception at utterance level. (Data)

Multimodal Deception Detection in Dialogues (pdf, data, abstract, BibTeX)
Felix Soldner, Verónica Pérez-Rosas, Rada Mihalcea
NAACL-HLT, 2019

Open-Domain Deception

August 27, 2015

This is a crowdsourced deception dataset consisting of short open domain truths and lies from 512 users. Seven lies and seven truths are provided for each user. The dataset also includes user's demographic information, such as gender, age, country of origin, and education level. (download)

Experiments in Open Domain Deception Detection (pdf, abstract, BibTeX)

Note: If the Dataset Download fails on Chrome Browser, use Firefox Browser or below link.

Above Dataset uploaded on G-Drive : https://drive.google.com/drive/folders/1v1LDcvGZhBV-Ffq1uPfeyAT2NVrtORFf?usp=sharing

Tools to be used.

Programming Language: Python

Frameworks : TensorFlow, Keras.

Implementation Environment: Jupyter Notebook