Fake News Data: ()

Assumptions:

1. **Removed the rows that didn't have a label associated with it.**
2. **Used only the text to develop the classifier**

Resources: MATLAB 2017b (Neural Network Toolbox and Parallel Computing Toolbox), NVIDIA K-40 GPU (for training)

Initial set up

```
clear

% Set the parameters below as true if you want to train the model.
doTrain = true;
doTest = true;
```

Loading the Dataset

```
% filename = "fake_or_real_news.xlsx";
% data = readtable(filename,'TextType','string');
% save('fakeNewsData.mat')
  load('fakeNewsData.mat');
```

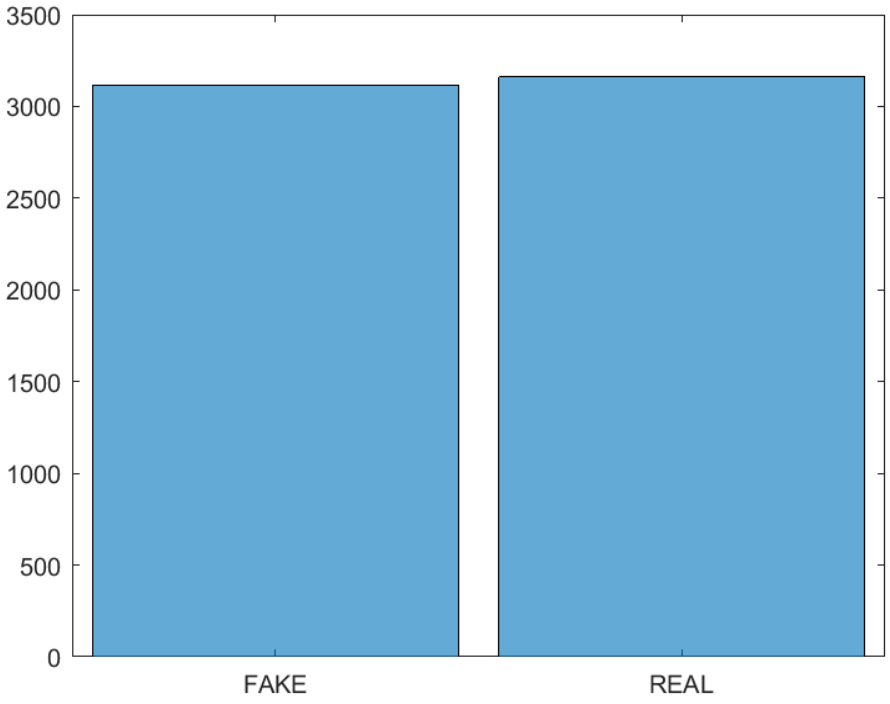
Consider the rows where the dataset is labeled as FAKE or REAL.

Remove the rows where the text is blank

```
idxLabel = find(data.label ~= 'FAKE' & data.label ~= 'REAL');
data(idxLabel,:) = [];

idxEmpty = strlength(data.text) == 0;
data(idxEmpty,:) = [];

data.label = categorical(data.label);
```

visualizing the distribution of the classes. There appears to be non class imbalance

```
h = histogram(data.label);
```



Partitioning the dataset

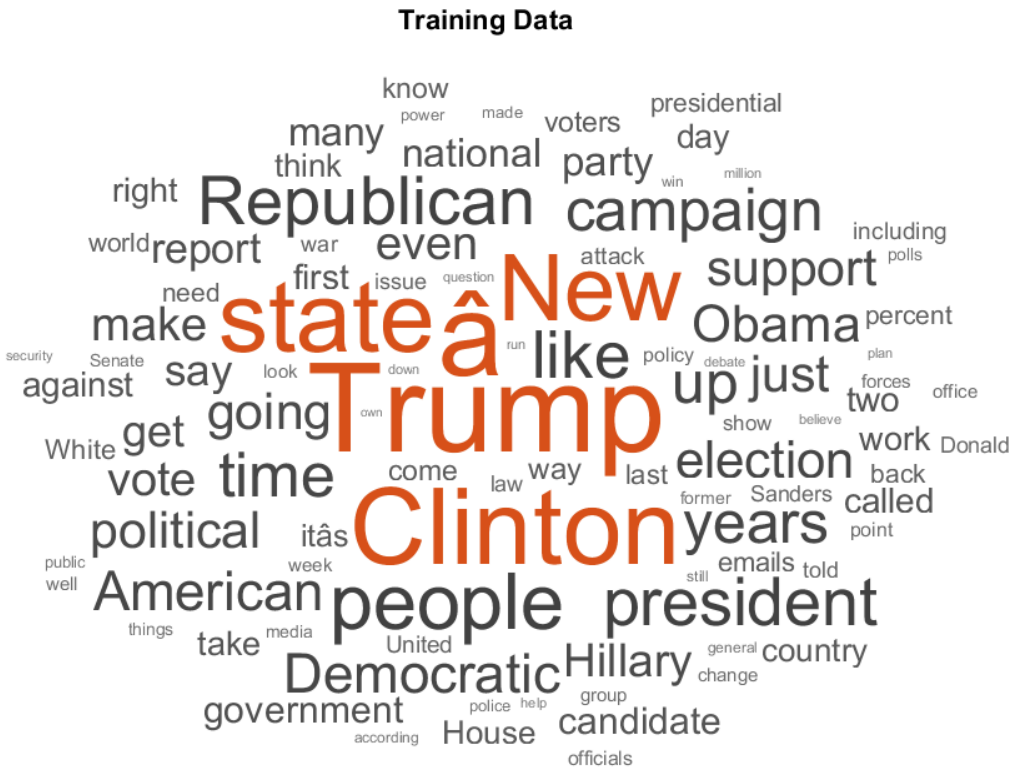
```
cvp = cvpartition(data.label,'Holdout',0.1);
dataTrain = data(training(cvp),:);
dataTest = data(test(cvp),:);
```

Partitioning into training and test sets

```
textDataTrain = dataTrain.text;
textDataTest = dataTest.text;
YTrain = dataTrain.label;
YTest = dataTest.label;
```

visualize training data using a word cloud

```
figure
wordcloud(textDataTrain);
title("Training Data")
```



Little Preprocessing

```
textDataTrain = erasePunctuation(textDataTrain);
textDataTrain = lower(textDataTrain);
documentsTrain = tokenizedDocument(textDataTrain);
```

Word Embedding

```
tic;
embeddingDimension = 100;
embeddingEpochs = 50;

if doTrain
    emb = trainWordEmbedding(documentsTrain, ...
        'Dimension',embeddingDimension, ...
        'NumEpochs',embeddingEpochs, ...
        'Verbose',1)
else
    load('emb.mat');
end
```

```
Training: 100% Loss: 1.64922  Remaining time: 0 hours 0 minutes.
emb =
  wordEmbedding with properties:

     Dimension: 100
```

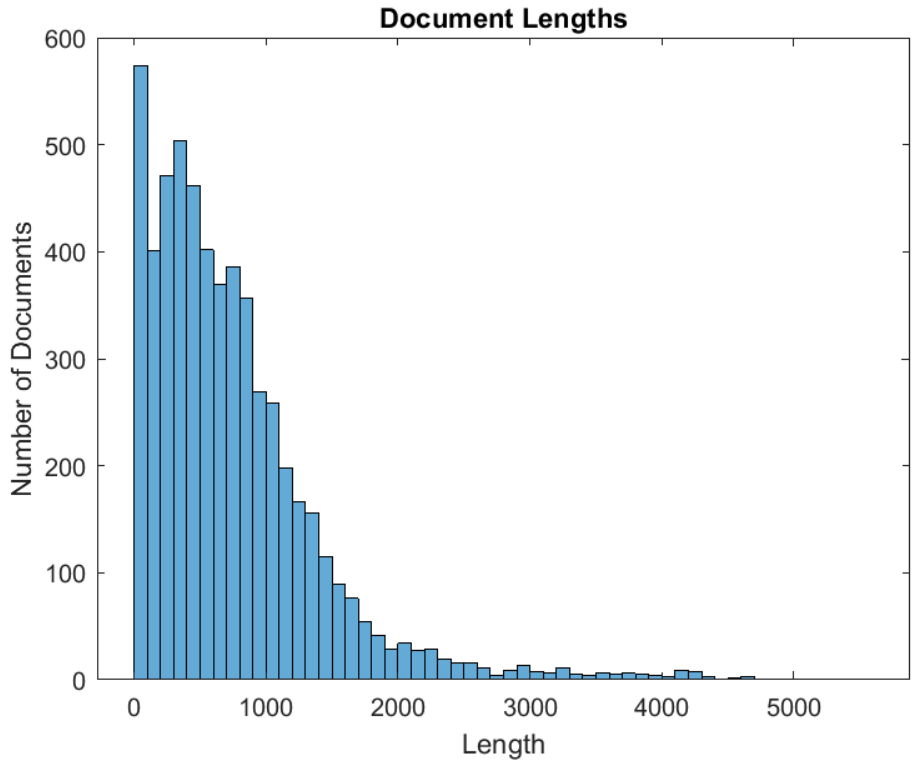
```
Vocabulary: [1×28987 string]
```

```
toc
```

```
Elapsed time is 732.518802 seconds.
```

length of docs

```
documentLengths = doclength(documentsTrain);
figure
histogram(documentLengths)
title("Document Lengths")
xlabel("Length")
ylabel("Number of Documents")
```



Since most of the docs have tokens of size 1200, padding with seq length of 1200

```
sequenceLength = 1200;
documentsTruncatedTrain = docfun(@(words) words(1:min(sequenceLength,end)),documentsTrain);
save('emb_1200')
```

Converting the doc to sequences to be fed into the LSTM

```
XTrain = doc2sequence(emb,documentsTruncatedTrain);
```

```
Starting parallel pool (parpool) using the 'local' profile ...
connected to 12 workers.
```

Left Padding

```
for i = 1:numel(XTrain)
    XTrain{i} = leftPad(XTrain{i},sequenceLength);
end
XTrain(1:5)';
```

Train Network

```
inputSize = embeddingDimension;
outputSize = 100;
numClasses = numel(categories(YTrain));

layers = [ ...
    sequenceInputLayer(inputSize)
    lstmLayer(outputSize,'OutputMode','last')
    fullyConnectedLayer(numClasses)
    softmaxLayer
    classificationLayer]
```

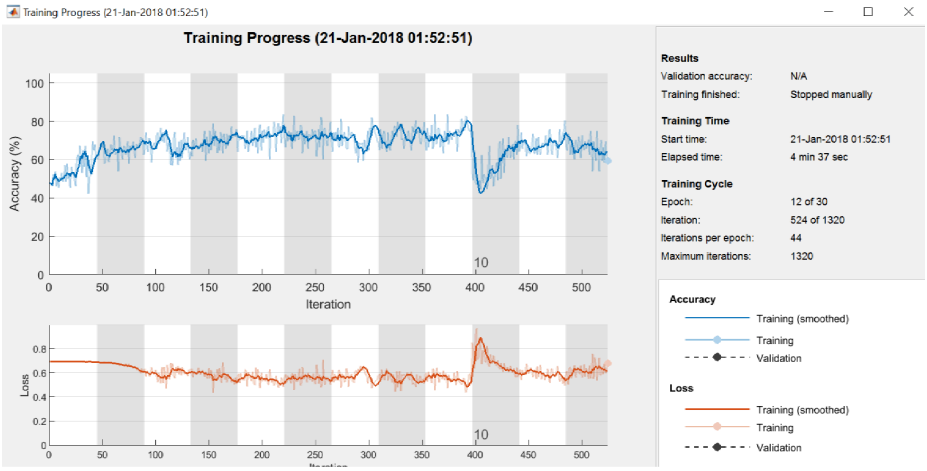
```
layers =
  5x1 Layer array with layers:

     1   ''   Sequence Input          Sequence input with 100 dimensions
     2   ''   LSTM                    LSTM with 100 hidden units
     3   ''   Fully Connected         2 fully connected layer
     4   ''   Softmax                 softmax
     5   ''   Classification Output   crossentropyex
```

```
options = trainingOptions('sgdm', ...
    'InitialLearnRate',0.01, ...
    'Plots','training-progress', ...
    'Verbose',1)
```

Train Network (training with default options)

```
if doTrain
    net = trainNetwork(XTrain,YTrain,layers,options);
    save('firstPassNetwork.mat')
else
    load('lstmFake_firstPass.mat');
    imshow('firstPassNetwork.PNG');
```

```
end
```



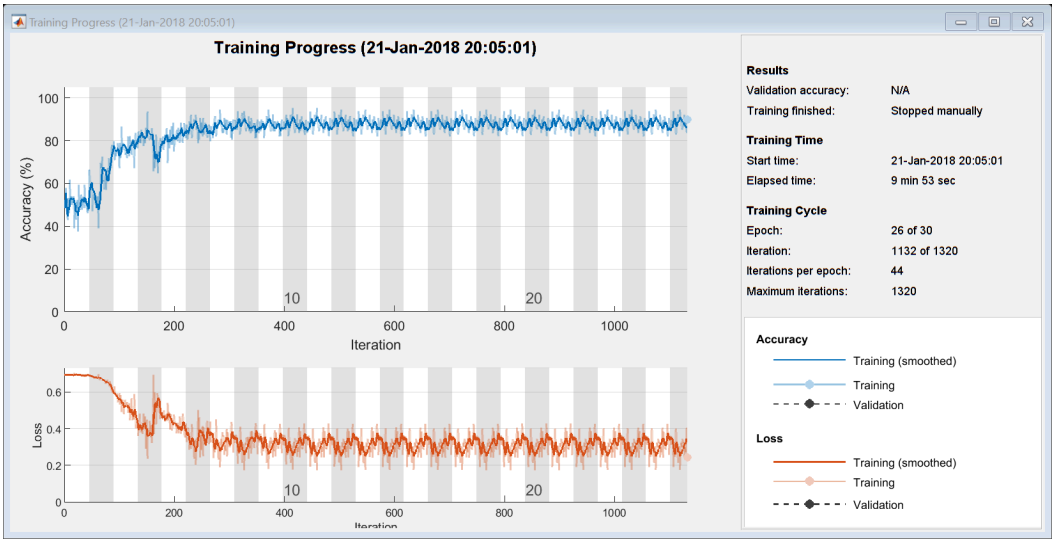
Runnning the network on default hyperparameters yielded low accuracy with drops in training progress. This suggests that the Initial learning rate be adjusted, and possibly, reduced after every nth epoch so that the objective func reaches a global minima and not diverge. Hence, adding a drop factor of 0.2 after every 2 epochs so that the steps for the learning rate become smaller with time

```
options = trainingOptions('sgdm', ...
    'LearnRateSchedule','piecewise',...
    'InitialLearnRate',0.05, ...
    'LearnRateDropFactor',0.2,...
    'LearnRateDropPeriod',2,...
    'Shuffle','never',...
    'Plots','training-progress', ...
    'Verbose',1)
```

```
options =
  TrainingOptionsSGDM with properties:

                    Momentum: 0.9000
           InitialLearnRate: 0.0500
    LearnRateScheduleSettings: [1×1 struct]
           L2Regularization: 1.0000e-04
      GradientThresholdMethod: 'l2norm'
            GradientThreshold: Inf
                   MaxEpochs: 30
               MiniBatchSize: 128
                     Verbose: 1
             VerboseFrequency: 50
               ValidationData: []
          ValidationFrequency: 50
           ValidationPatience: 5
                     Shuffle: 'never'
              CheckpointPath: ''
        ExecutionEnvironment: 'auto'
                  WorkerLoad: []
                   OutputFcn: []
                       Plots: 'training-progress'
             SequenceLength: 'longest'
```

```
    SequencePaddingValue: 0
```

```
if doTrain
    net = trainNetwork(XTrain,YTrain,layers,options);
    save('thirdPassNetwork.mat')
else
    load('lstmFake_secondPass_88.mat');
    imshow(('secondPassNetwork.PNG'));
end
```




```
Training on single GPU.
|=========================================================================================|
|  Epoch  |  Iteration  |  Time Elapsed  |  Mini-batch  |  Mini-batch  |  Base Learning  |
|         |             |   (hh:mm:ss)   |   Accuracy   |     Loss     |      Rate       |
|=========================================================================================|
|       1 |           1 |       00:00:01 |       52.34% |       0.6931 |          0.0500 |
|       2 |          50 |       00:00:26 |       57.03% |       0.6825 |          0.0500 |
|       3 |         100 |       00:00:52 |       67.19% |       0.5932 |          0.0100 |
|       4 |         150 |       00:01:18 |       82.81% |       0.4485 |          0.0100 |
|       5 |         200 |       00:01:44 |       78.13% |       0.4621 |          0.0020 |
|       6 |         250 |       00:02:11 |       85.16% |       0.3944 |          0.0020 |
|       7 |         300 |       00:02:38 |       89.06% |       0.3042 |          0.0004 |
|       8 |         350 |       00:03:05 |       87.50% |       0.3628 |          0.0004 |
|      10 |         400 |       00:03:32 |       86.72% |       0.3637 |       8.0000e-05 |
|      11 |         450 |       00:03:59 |       89.84% |       0.2624 |       1.6000e-05 |
|      12 |         500 |       00:04:25 |       90.63% |       0.2145 |       1.6000e-05 |
|      13 |         550 |       00:04:51 |       85.94% |       0.3001 |       3.2000e-06 |
|      14 |         600 |       00:05:17 |       86.72% |       0.3004 |       3.2000e-06 |
|      15 |         650 |       00:05:44 |       88.28% |       0.3031 |       6.4000e-07 |
|      16 |         700 |       00:06:10 |       82.81% |       0.4268 |       6.4000e-07 |
|      18 |         750 |       00:06:36 |       84.38% |       0.3277 |       1.2800e-07 |
|      19 |         800 |       00:07:02 |       87.50% |       0.3284 |       2.5600e-08 |
|      20 |         850 |       00:07:29 |       89.84% |       0.2725 |       2.5600e-08 |
|      21 |         900 |       00:07:55 |       89.84% |       0.2792 |       5.1200e-09 |
|      22 |         950 |       00:08:21 |       86.72% |       0.3289 |       5.1200e-09 |
|      23 |        1000 |       00:08:47 |       89.84% |       0.2418 |       1.0240e-09 |
|      24 |        1050 |       00:09:14 |       85.94% |       0.3474 |       1.0240e-09 |
|      25 |        1100 |       00:09:37 |       85.16% |       0.3342 |       2.0480e-10 |
|      26 |        1132 |       00:09:53 |       89.84% |       0.2418 |       2.0480e-10 |
|=========================================================================================|
Warning: Variable 'XTrain' was not saved. For variables larger than 2GB use MAT-file version 7.3 or later.
```

testing

```
textDataTest = erasePunctuation(textDataTest);
textDataTest = lower(textDataTest);
documentsTest = tokenizedDocument(textDataTest);
```

Convert the docs to seq

```
documentsTruncatedTest = docfun(@(words) words(1:min(sequenceLength,end)),documentsTest);
XTest = doc2sequence(emb,documentsTruncatedTest);
```

```
Starting parallel pool (parpool) using the 'local' profile ...
connected to 12 workers.
```

```
for i=1:numel(XTest)
    XTest{i} = leftPad(XTest{i},sequenceLength);
end
XTest(1:5)
```

```
ans = 1×5 cell array
    {100×1200 single}    {100×1200 single}    {100×1200 single}    {100×1200 single}    {100×1200 single}
```

Inference

```
YPred = classify(net,XTest);
```

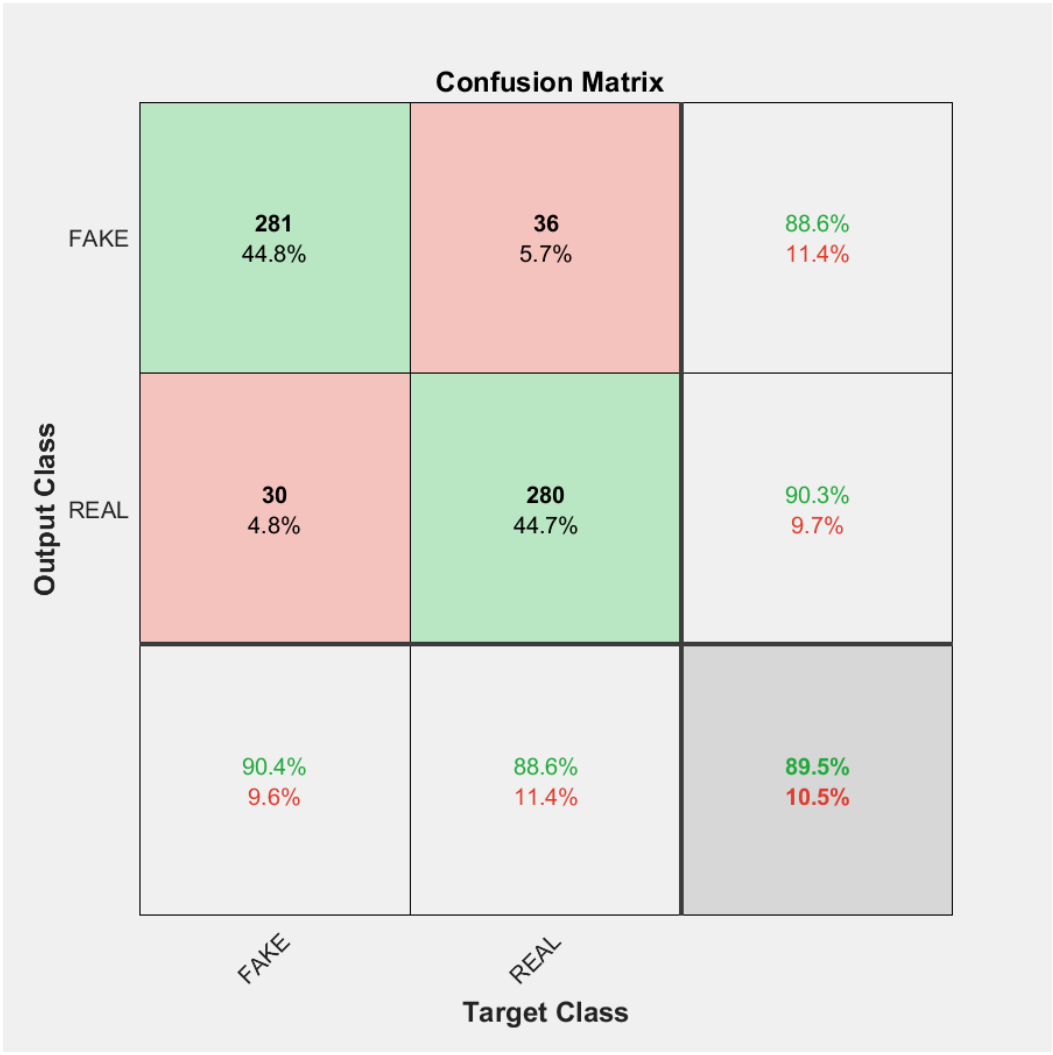
accuracy

```
 accuracy = sum(YPred == YTest)/numel(YPred)
```

```
accuracy = 0.8947
```

Plot confusion

```
plotconfusion(YTest,YPred);
```


Confusion Matrix
FAKE
REAL
Output Class
281
44.8%
36
5.7%
88.6%
11.4%
30
4.8%
280
44.7%
90.3%
9.7%
90.4%
9.6%
88.6%
11.4%
89.5%
10.5%
FAKE
REAL
Target Class