Sentiment Analysis of Twitter Posts

Michael Scheid
michael.scheid@student.csulb.edu
CECS 590
12/6/2018

The goal of this project is to construct and analyze Long-Short Term Memory, LSTM, neural networks for the task of ranking Twitter content on a three-point sentiment scale. This project contains three major parts: natural language processing, neural network training, and sentiment analysis of 2018 California General Election data. A neural network with only 1 LSTM layer trained faster and obtained a greater accuracy when compared to networks with multiple LSTM layers. The trained LSTM network is used to categorize 1000 tweets from 18 candidates from the California 2018 General Election.

Outline

Introduction
Part 1: NLP Challenges and techniques p. 1 - 2
Part 2: LSTM: Overview p. 2 - 3
Methods/Results
Part 1: NLP p. 3 - 4
Part 2: LSTM p. 4 - 7
Part 3: 2018 California General Election Data p. 8 - 10
Conclusion p. 10 - 11
Bibliography p. 11
Appendix A: Matlab and Python Code p. 12 - 21

Introduction

Part 1: NLP

The field of natural language processing, contains various terminology which will be used in describing this project. In a popular survey paper, Pang and Lee review a the body of work that covers “the computational treatment of […] opinion, sentiment, and subjectivity in text. Such work has come to be known as opinion mining, sentiment analysis, and/or subjectivity analysis.“ [6] (Pang and Lee, p. 8). Sentiment analysis is a natural language processing task of extracting a broad sentiment of a document. The three-point sentiment scale used for this project was negative, neutral, and positive. A corpus is just a collection of documents. A document is the unit of text which the system will use. [7] (Manning, Raghavan & Schütze, p.4). A twitter corpus is used for training the neural network. The training corpus used for this project contained 20,000 individual tweet, or documents, each labeled with a sentiment. The test corpus contained 7,124 documents labeled.

There are a number of challenges in performing sentiment analysis on a document. Pang and Lee describe one of the key challenges relevant to sentiment analysis:

> "In general, sentiment and subjectivity are quite context-sensitive […] even the exact same expression can indicate different sentiment in different domains. For example, "go read the book" most likely indicates positive sentiment for book reviews, but negative sentiment for movie reviews. (This example was furnished to us by Bob Bland.)" [6] (Pang and Lee, p.21)

Pang and Lee also describe how the granularity of a document can also impact the process extraction of sentiment. Extended twitter documents are limited to 280 characters, but this project limited the document length to the Twitter platforms' original 140 characters limit.

The core of this project is to analyze LSTM neural networks in the task of sentiment analysis; thus, the most important natural language processing task is converting the tweet document to a form that can be used as the input of a neural network. Neural networks perform matrix operations on numeric input therefore we must convert each tweet document into a numeric vector representation. The vector representation of words must contain contextual information if we wish to classify the sentiment accurately. Word2Vec and GloVe are two algorithms used for creating the vector representation, also known as the embedding, of words which contains the required contextual information. Each algorithm uses a neural network that learns to maximize the probability of context words related given a target word. The advantage of GloVe is that it relies of only relies on the target word and co-word frequencies whereas word2Vec uses a much more computationally complex technique of passing a sliding window over the text of each document to predict the surrounding words.[9] (Manning, C., Socher, R),[10](Socher, R.). In this project, I used the pre-trained GloVe word vectors which were trained on 2 billion tweets and are of length 100.

Part 2: LSTM

Long-Short Term Memory, LSTM, networks are a type of Recurrent Neural Networks, RNN, which processes a sequence of data in discrete time steps. The RNN architecture contains a memory cell that stores the weighted input of the current time step which is used as an additional input for subsequent time steps. LSTM networks contain the same architecture as RNNs, but the cell contains additional components.

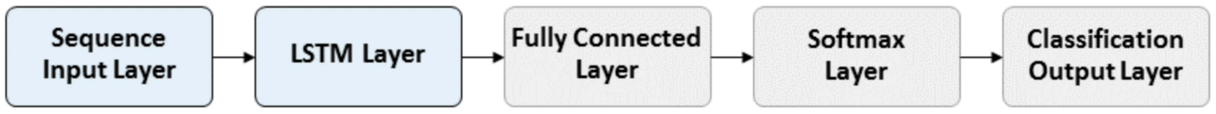


Figure 1: Layers used in LSTM neural network for sentiment classification [3](MATLAB Documentation)

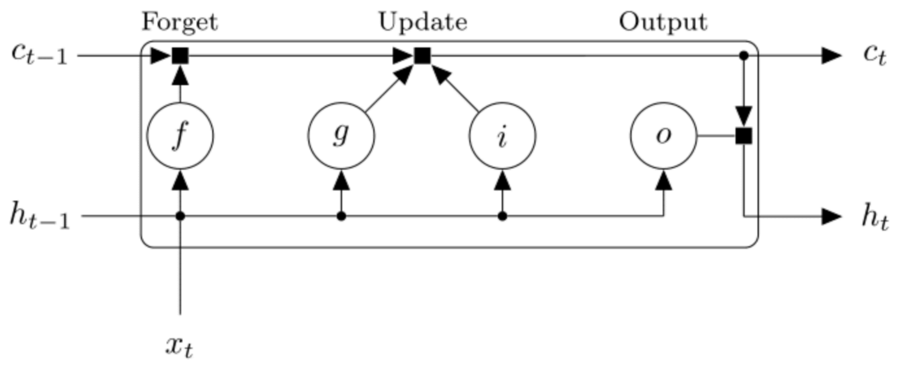


Figure 2: LSTM Gate Diagram [3](MATLAB Documentation)

$$\begin{aligned}
f_t(x_t, h_{t-1}) &= \sigma(W_{forget} \times x_t + R_{forget} \times h_{t-1} + bias_{forget}) && (1)\\
i_t(x_t, h_{t-1}) &= \sigma(W_{input} \times x_t + R_{input} \times h_{t-1} + bias_{input}) && (2)\\
g_t(x_t, h_{t-1}) &= \tanh(W_{cellstate} \times x_t + R_{cellstate} \times h_{t-1} + bias_{cellstate}) && (3)\\
C_t &= i_t \times g_t + f_t \times C_{t-1} && (4)\\
o_t(x_t, h_{t-1}) &= \sigma(W_{output} \times x_t + R_{output} \times h_{t-1} + bias_{output}) && (5)\\
h_t &= o_t \times \tanh(C_t) && (6)
\end{aligned}$$

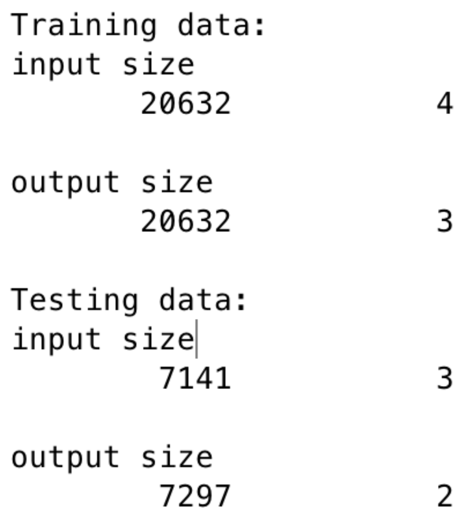
Figure 3: LSTM Gate Equations. Forget, input, and output gates: f, i and o. C is the cell state. [3](MATLAB Documentation)

The LSTM cell shown in figure 2 contains three gates that control the flow and storage of information in the cell. The forget and input gates alter the memory or cell state while the output gate alters the flow coming out of the cell. All the gates have two inputs which are the input at the current time step and the LSTM output of the previous time step. Furthermore the gates values are squished to be between 0 and 1 by the sigma activation function. An important feature of RNNs and LSTM networks is that information at a previous time step should be capable of negatively or positively impacting the subsequent time steps; therefore the tanh activation function is used to allow time step output, h, to contribute between -1 and 1.

Methods

Part 1: NLP

The training and testing corpus were obtained from SemEval-2017 task 4 subtask A. SemEval is an annual international workshop on semantic evaluation. The SemEval organizers release labeled datasets and corresponding tasks for groups to use. The data set used contained roughly 20,000 english tweets that were labeled with a negative, neutral, or positive sentiment. On the following page, figure 4 shows the exact input and output size. All the data was read from the files and stored in Matlab tables.



```
Training data:
input size
       20632          4

output size
       20632          3

Testing data:
input size
        7141          3

output size
        7297          2
```

Figure 4 on the following page shows the actual input and output size.

The test set was also from the SemEval-2017 task 4 subtask A. The test set was split into two different files. The first file contained the IDs and tweets while the second file contained the IDs and sentiment. Figure 4 shows that the number of entries in the test set files did not match. Resolving that issue required removing entries who's IDs were not in both tables.

As mentioned in the introduction, the tweet text was covered into a form suitable for processing by the LSTM network. The text is converted to all lower case. All punctuation and links were stripped from the text. Stop words which are commonly used words that wouldn't effect the sentiment were removed. Finally, the text was tokenized. The tokenization process breaks the text string into an array of single words. Each word is then covered into a vector using a pertained GloVe file to obtain the vector representation. The vector is then padded with zeros to create a standard sized input for the LSTM. The vector representation of the tweets were stored in a stuct.

Part 2: LSTM

The LSTM network contained the structure shown in figure 1. The word vectors, sentiment labels and network options were passed into the trainNetwork function. Training the network even when using up to 4 LSTM was rather quick because the tokenized documents were quite small. The next few pages show a select number of training runs.

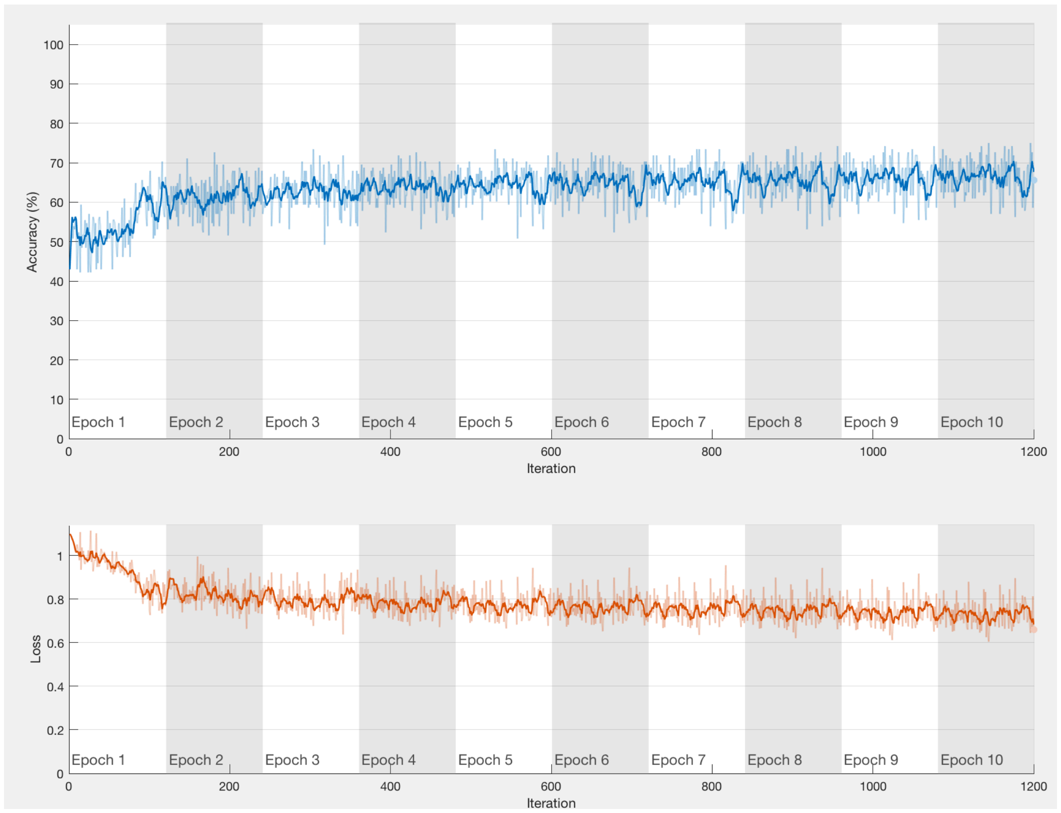


Figure 5. Training with 1 LSTM layer. Initial learning rate of 0.05. Test accuracy: 64.56%

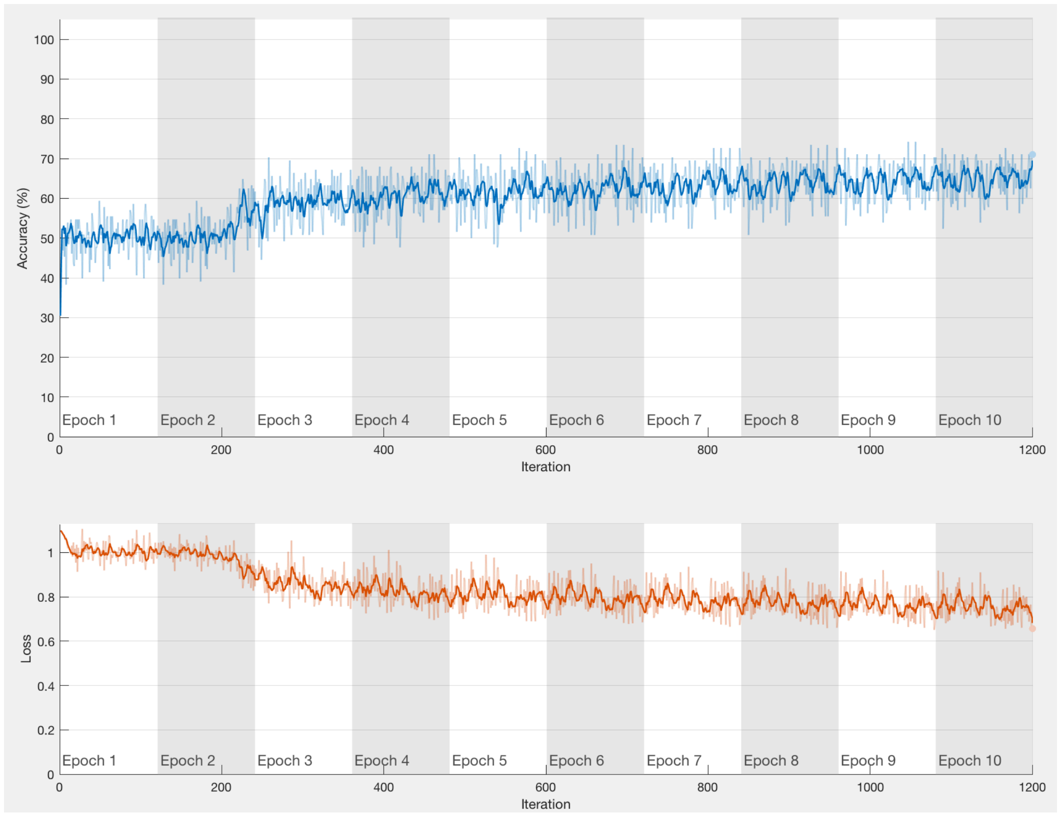


Figure 6. Training with 2 LSTM layers. Initial learning rate of 0.05. Test accuracy: 62.93%

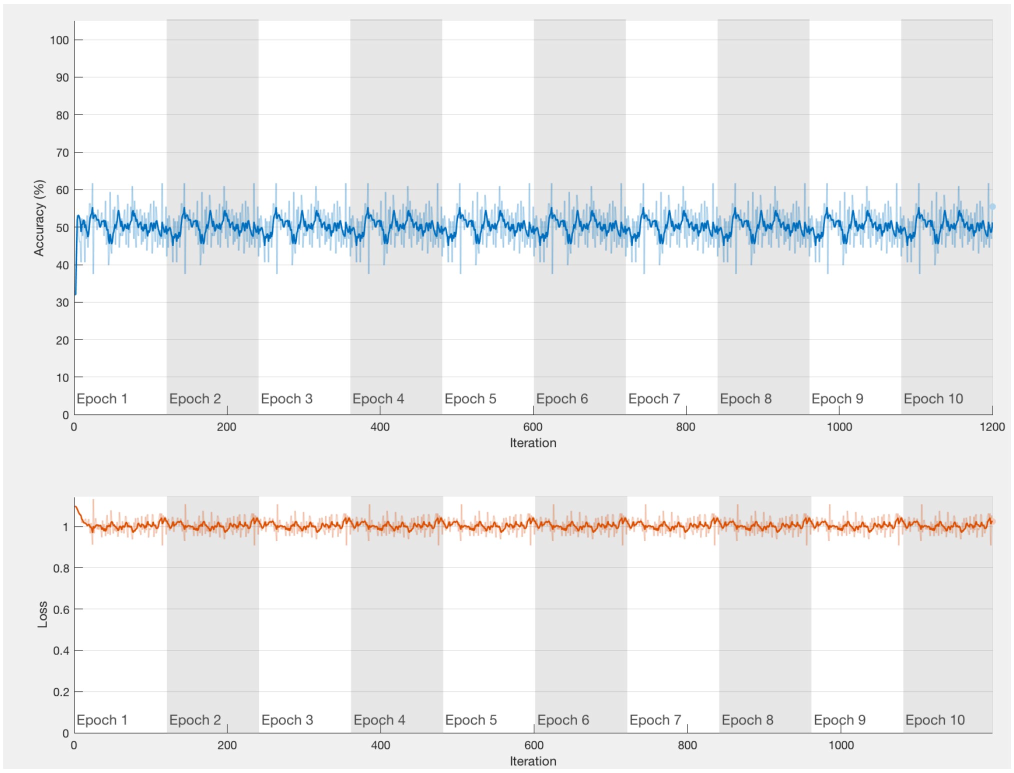


Figure 7. Training with 4 LSTM layers. Initial learning rate of 0.05. Test accuracy: 49.05%

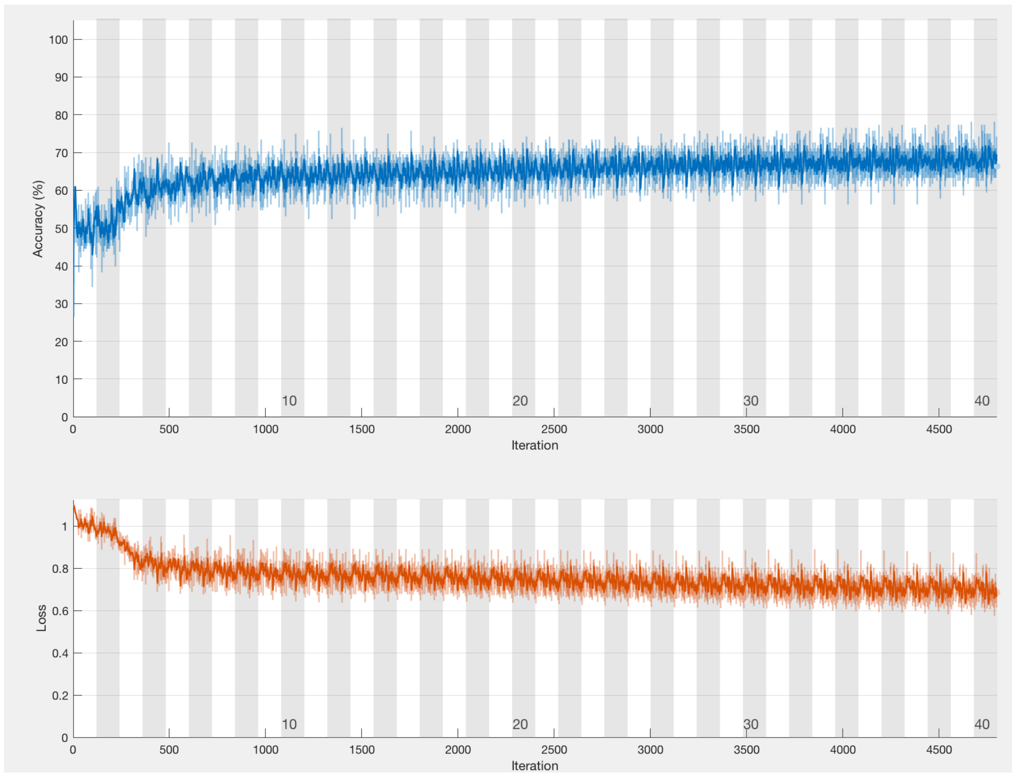


Figure 8. Training with 1 LSTM layer. Initial learning rate of 0.01. Increased number of epochs. Test accuracy: 63.97%

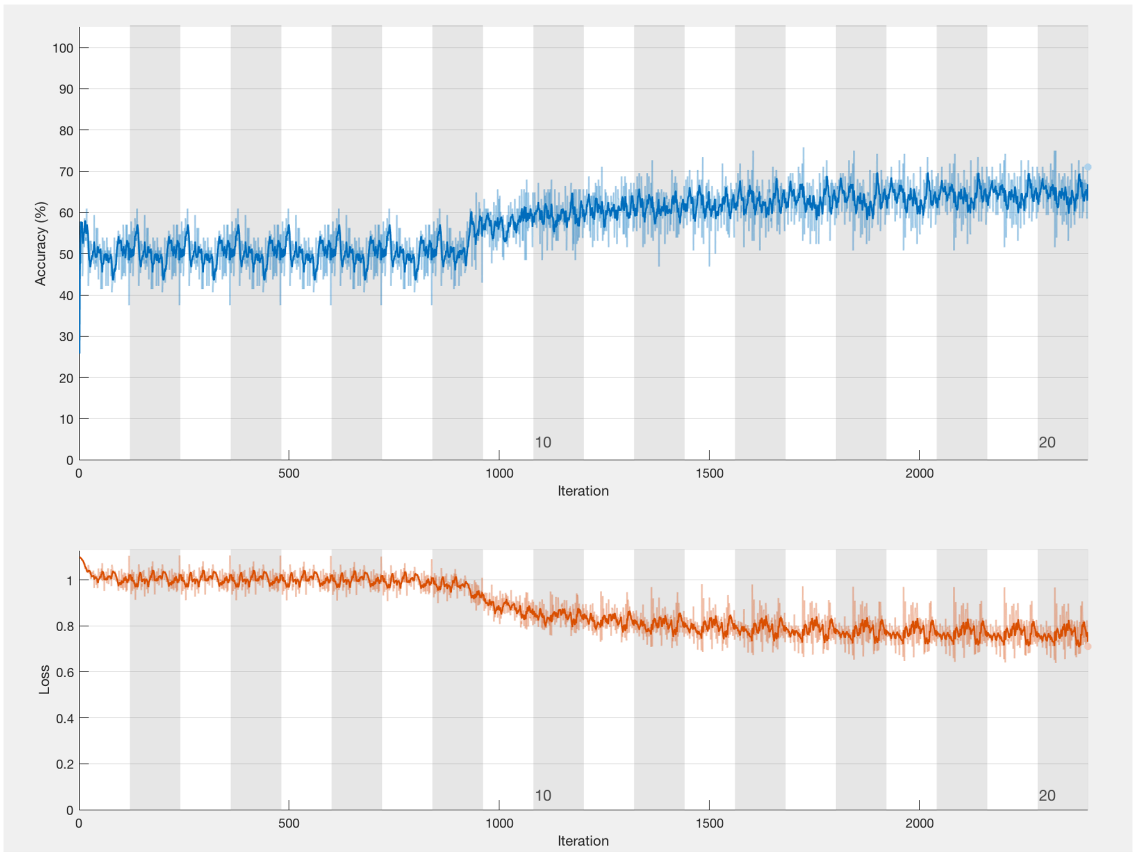
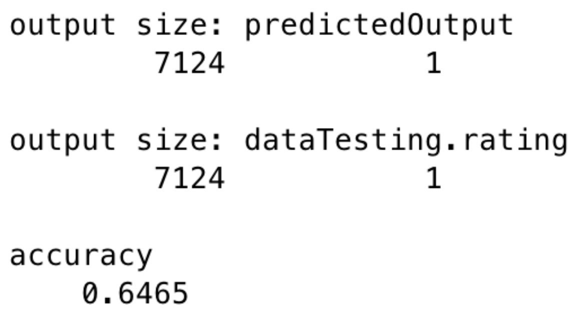


Figure 9. Training with 2 LSTM layers. Initial learning rate of 0.01. Increased number of epochs. Test accuracy: 63.97%



```
output size: predictedOutput
        7124              1

output size: dataTesting.rating
        7124              1

accuracy
    0.6465
```

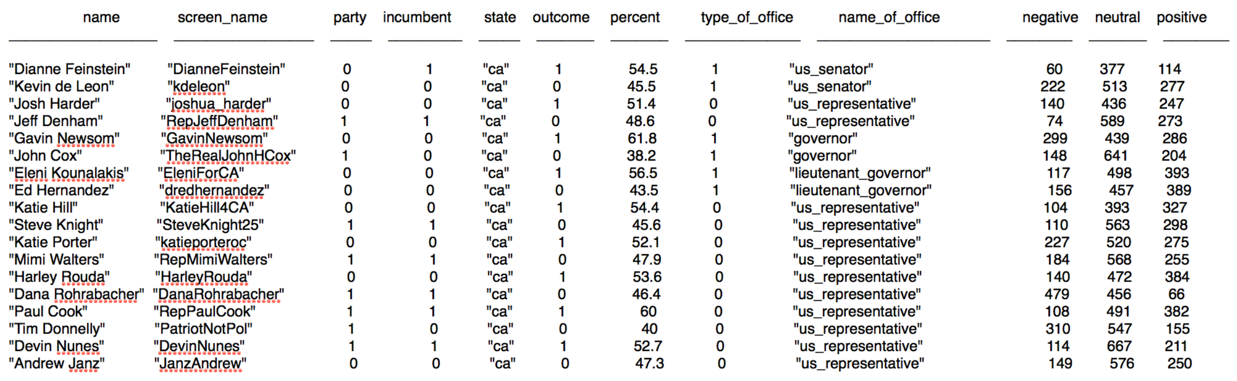
Figure 10. Example output shown after figure 5 was produced.

Figure 1 and 10 show the best results obtained after varying the learning rate, number of LSTM layers, number of nodes in the hidden layer.

Part 3: 2018 California General Election Data

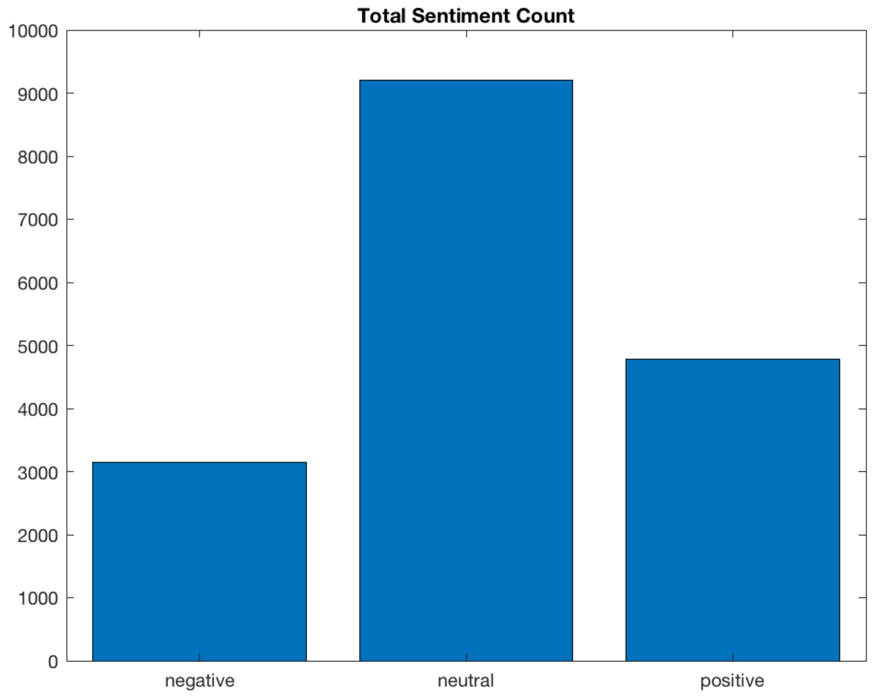
I wanted to analyze the sentiment of twitter data from around the 2018 General Election. I limited the scope of the of the investigation to the 2018 California General Election because of time constraints. The election data available on the New York Times website allowed me to enter the real names, percentages, and outcome from 9 races in California. I used Google to look up the twitter screen names for each of the 18 candidates. The Twitter API allows a developer to download tweets from a user by username. A small python application aided writing the election information to a file. Another small python application read the file containing the election information and download the past 1000 tweets from each of the candidates' Twitter timeline. The tweets for each candidate were saved to a file.

Next, raw twitter data was loaded into a Matlab struct so that the same preprocessing, tokenization, vector transformation methods could convert the data into an acceptable form for the LSTM network. Finally the sentiment of each tweet was predicted using the LSTM network. The sentiment count for each candidate's tweets were appended to the table containing the election data.

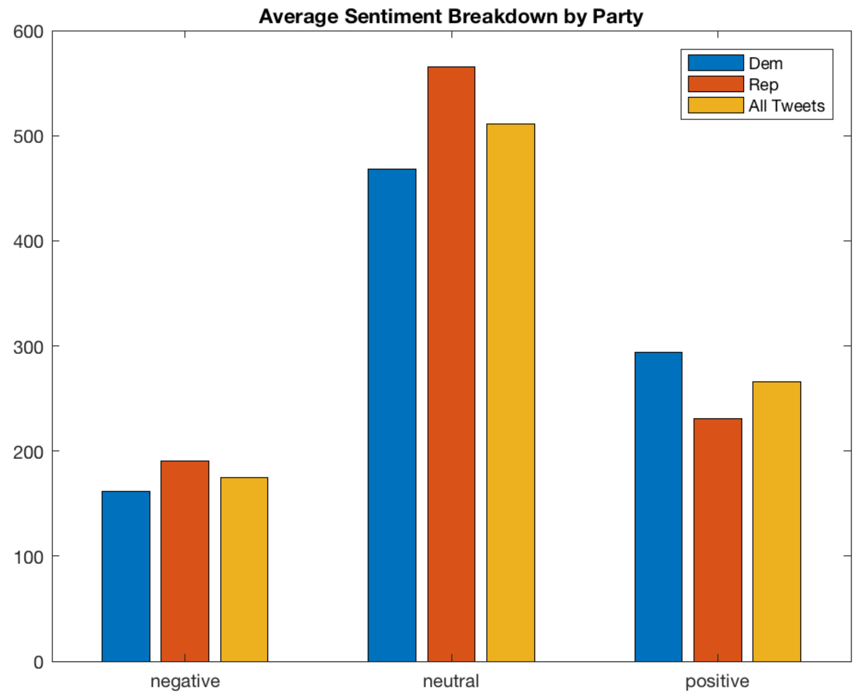


name	screen_name	party	incumbent	state	outcome	percent	type_of_office	name_of_office	negative	neutral	positive
"Dianne Feinstein"	"DianneFeinstein"	0	1	"ca"	1	54.5	1	"us_senator"	60	377	114
"Kevin de Leon"	"kdeleon"	0	0	"ca"	0	45.5	1	"us_senator"	222	513	277
"Josh Harder"	"joshua_harder"	0	0	"ca"	1	51.4	0	"us_representative"	140	436	247
"Jeff Denham"	"RepJeffDenham"	1	1	"ca"	0	48.6	0	"us_representative"	74	589	273
"Gavin Newsom"	"GavinNewsom"	0	0	"ca"	1	61.8	1	"governor"	299	439	286
"John Cox"	"TheRealJohnHCox"	1	0	"ca"	0	38.2	1	"governor"	148	641	204
"Eleni Kounalakis"	"EleniForCA"	0	0	"ca"	1	56.5	1	"lieutenant_governor"	117	498	393
"Ed Hernandez"	"dredhernandez"	0	0	"ca"	0	43.5	1	"lieutenant_governor"	156	457	389
"Katie Hill"	"KatieHill4CA"	0	0	"ca"	1	54.4	0	"us_representative"	104	393	327
"Steve Knight"	"SteveKnight25"	1	1	"ca"	0	45.6	0	"us_representative"	110	563	298
"Katie Porter"	"katieporteroc"	0	0	"ca"	1	52.1	0	"us_representative"	227	520	275
"Mimi Walters"	"RepMimiWalters"	1	1	"ca"	0	47.9	0	"us_representative"	184	568	255
"Harley Rouda"	"HarleyRouda"	0	0	"ca"	1	53.6	0	"us_representative"	140	472	384
"Dana Rohrabacher"	"DanaRohrabacher"	1	1	"ca"	0	46.4	0	"us_representative"	479	456	66
"Paul Cook"	"RepPaulCook"	1	1	"ca"	1	60	0	"us_representative"	108	491	382
"Tim Donnelly"	"PatriotNotPol"	1	0	"ca"	0	40	0	"us_representative"	310	547	155
"Devin Nunes"	"DevinNunes"	1	1	"ca"	1	52.7	0	"us_representative"	114	667	211
"Andrew Janz"	"JanzAndrew"	0	0	"ca"	0	47.3	0	"us_representative"	149	576	250

Figure 11: The data in the candidate table after predicting the sentiment.



Figures 12: Average sentiment predicted by the LSTM network with 1 LSTM layer.



Figures 13: Average sentiment of the tweets from the candidate arranged by party. Predicted by the LSTM network with 1 LSTM layer.

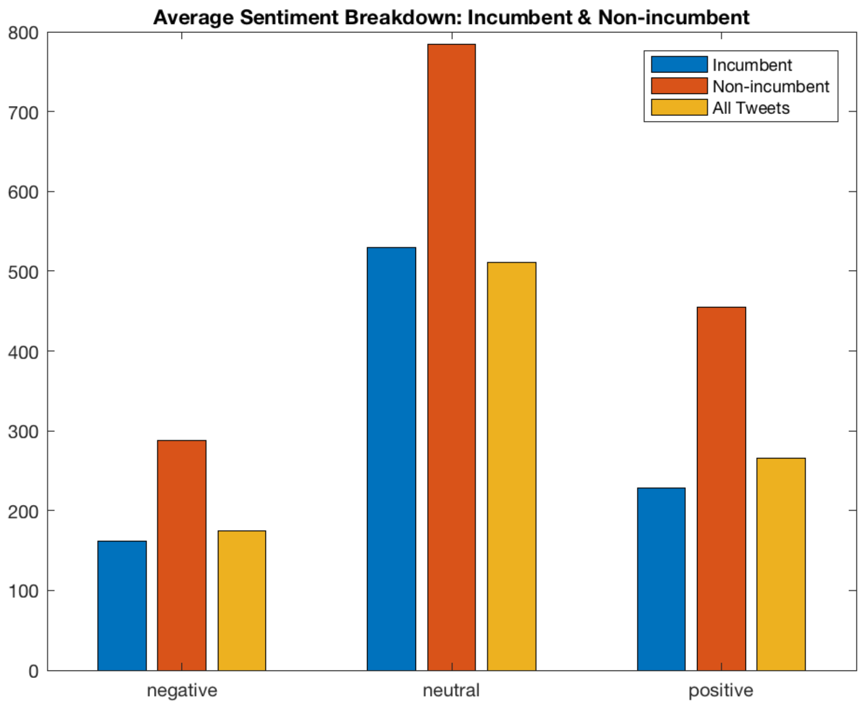


Figure 14 Average sentiment of the tweets from incumbents and non-incumbents. Predicted by the LSTM network with 1 LSTM layer.

Conclusion

The network that performed the best contained only 1 LSTM layer. The accuracy of predicting the test data was 63.65%. The accuracy of 63.65% is in line with the results shown in Table 6 column 3 of SemEval-2017 Task 4: Sentiment Analysis in Twitter. [2](Rosenthal, Farra & Nakov, 2017, p.). There are other measures that can be used to gauge the performance of the LSTM at the task of sentiment analysis. My single layer LSTM performed well, but it would be interesting to see how well it performed using other measures. For example average recall was the primary measure SemEval used to compare systems.

The project could have been improved by using Matlab R2018b. The ability to plot validation data when training LSTM networks was not introduced until Matlab R2018b. Unfortunately, I had already downloaded a free demo of the Text Analytics Toolbox using Matlab R2017b for this project. I made the decision to continue with Matlab R2017b rather than updating to R2018b and risk being unable to complete the project. That decision necessitated many more training trials to while varying the number of epochs and test the accuracy.

Twitter posts contained an upper limit of 140 characters for the majority of the platform's existence, but the upper limit was recently doubled. This project did not utilize the new upper limit. The short document length should result in a high information density which should make extracting the sentiment easier compared to long texts. The accuracy of classifying the sentiment could be improved by using full extended tweets.

Bibliography

[1]: Miedema, F., Bhulai, S, (2018, August 1). Sentiment Analysis with Long Short-Term Memory Networks. Retrieved from https://beta.vu.nl/nl/Images/werkstuk-miedema_tcm235-895557.pdf

[2]: Rosenthal, S., Farra, N., Nakov, P. (2017). SemEval-2017 Task 4: Sentiment Analysis in Twitter. Retrieved from http://www.aclweb.org/anthology/S17-2088

[3]: NA. (nd). MATLAB Documentation: Long Short-Term Memory Networks. Retrieved from https://www.mathworks.com/help/deeplearning/ug/long-short-term-memory-networks.html

[4]: NA. (nd). MATLAB Documentation: Classify Text Data Using Deep Learning. Retrieved from https://www.mathworks.com/help/textanalytics/ug/classify-text-data-using-deep-learning.html;jsessionid=198f40443de068e7330b1e76e04c

[5]: Olah, C. (2015, August 27). Understanding LSTM Networks. Retrieved from http://colah.github.io/posts/2015-08-Understanding-LSTMs/

[6]: Pang, B., Lee, L. (2008). Opinion Mining and Sentiment Analysis. Retrieved from http://www.cs.cornell.edu/home/llee/omsa/omsa-published.pdf

[7]: Manning, C., Raghavan, P,. Schütze, H. (2008). Introduction to Information Retrieval. Retrieved from https://nlp.stanford.edu/IR-book/

[8]: Harper, R. (2011, Spring). CS224n: Natural Language Processing with Deep Learning. Retrieved from http://web.stanford.edu/class/cs224n/

[9]: Manning, C., Socher, R. (2017, April 3). Lecture 2 Word Vector Representations: word2vec. Retrieved from https://www.youtube.com/watch?v=ERibwqs9p38

[10]: Socher, R. (2017, April 3). Lecture 3 | GloVe: Global Vectors for Word Representation. Retrieved from https://www.youtube.com/watch?v=ERibwqs9p38

Appendix A: Source Code

LSTM Training Matlab Code: (Requires Matlab's Text Analytics Toolbox):

lstm.m

```
% Load the data Semeval 2017

% Define the directory
trainingDir = "/Users/mjscheid/Desktop/590_Project/datasets/SemevalData_Full/Training/";
testingDir = "/Users/mjscheid/Desktop/590_Project/datasets/SemevalData_Full/Testing/";

% Define the file names
trainingDataInputFileName = "SemEval2017-task4-dev.subtask-A.english.INPUT.txt";
trainingDataOutputFileName = "twitter-2016test-A-English.txt";
testingDataInputFileName = "SemEval2017-task4-test.subtask-A.english.txt";
testingDataOutputFileName = "SemEval2017_task4_subtaskA_test_english_gold.txt";

% Define the full file path
trainingInputFile = trainingDir + trainingDataInputFileName;
trainingOutputFile = trainingDir + trainingDataOutputFileName;
testingInputFile = testingDir + testingDataInputFileName;
testingOutputFile = testingDir + testingDataOutputFileName;

% Build a table from the data from the text file
trainingInput = readtable(trainingInputFile,'TextType','string');
trainingOutput = readtable(trainingOutputFile,'TextType','string');
testingInput = readtable(testingInputFile,'TextType','string');
testingOutput = readtable(testingOutputFile,'TextType','string');

% I had an issue where the testing input and output weren't the same size
% Remove lines where the IDs don't match to ensure the data matches properly
idx_output  = ismember(testingOutput.Var1, testingInput.Var1);
testingOutput(~idx_output,:)=[];
idx_input  = ismember(testingInput.Var1, testingOutput.Var1);
testingInput(~idx_input,:)=[];

disp("testing data:");
disp("input size");
disp(size(testingInput));
disp("output size");
disp(size(testingOutput));


% Join the testing input and output into one table
dataTesting = testingInput;
dataTesting.Var2 = testingOutput.Var2;

% For some reason the training output table has a column of blank strings
% This line removes that empty column in trainingOutput
trainingOutput.Var3 = [];

% Add variable names to the tables
defult_vat_names = ["Var1","Var2","Var3"];
new_var_names = ["id","rating","tweet"];
```

```
for i = 1:3
   trainingInput.Properties.VariableNames{char(defult_vat_names(1,i))} = char(new_var_names(1,i));
   dataTesting.Properties.VariableNames{char(defult_vat_names(1,i))} = char(new_var_names(1,i));
   if i < 3
      trainingOutput.Properties.VariableNames{char(defult_vat_names(1,i))} = char(new_var_names(1,i));
   end
end

% Divide the dataset into Training, Validation sets
% cross-validation partition for data. https://www.mathworks.com/help/stats/cvpartition.html
% Train - 75%
% Validation = 25%
primaryPartition = cvpartition(trainingInput.rating,'Holdout',0.25);
dataTrain = trainingInput(primaryPartition.training,1:3);
dataValidation = trainingInput((primaryPartition.test),1:3);

% Preprocess the data
% Convert the tweet body of type string to of type tokenizedDocument
% See https://www.mathworks.com/help/textanalytics/ref/tokenizeddocument.html
dataTrain.tokenizedTweets = preprocessTweets(dataTrain.tweet);
dataValidation.tokenizedTweets = preprocessTweets(dataValidation.tweet);
dataTesting.tokenizedTweets = preprocessTweets(dataTesting.tweet);

dataTrain.rating = categorical(dataTrain.rating);
dataValidation.rating = categorical(dataValidation.rating);
dataTesting.rating = categorical(dataTesting.rating);

% Get the word embedding using pretrained GloVe not word2vec
wordEmbedding = downloadReadWordEmbedding(100);

% Remove empty tweets:
% These two lines are from the demo
% https://www.mathworks.com/matlabcentral/fileexchange/68264-classify-sentiment-of-tweets-using-deep-
learning
[~, idx] = removeEmptyDocuments(dataTrain.tokenizedTweets);
dataTrain(idx,:) = [];
%dataValidation
[~, idx] = removeEmptyDocuments(dataValidation.tokenizedTweets);
dataValidation(idx,:) = [];
%dataTesting
[~, idx] = removeEmptyDocuments(dataTesting.tokenizedTweets);
dataTesting(idx,:) = [];


% Use the word embedding to convert the tokenized tweets into vectors
% The helper function prepData is at the bottom of the file
dataTrain.wordVec = prepData(wordEmbedding,dataTrain.tokenizedTweets);
dataValidation.wordVec = prepData(wordEmbedding,dataValidation.tokenizedTweets);
dataTesting.wordVec = prepData(wordEmbedding,dataTesting.tokenizedTweets);


%build LSTM
inputSize = wordEmbedding.Dimension;
outputSize = 180;
numClasses = 3;

%{
   %Inital test setup
```

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


%https://www.mathworks.com/matlabcentral/answers/397588-how-is-it-possible-to-use-a-validation-set-with-a-
lstm
%Do you have R2018b? ValidationData is supported for sequence networks in R2018b.
%'ValidationData',{dataValidation.wordVec,dataValidation.rating}, ...
options = trainingOptions('sgdm',...
    'InitialLearnRate',0.05,...
    'Plots','training-progress',...
    'MaxEpochs',10,...
    'Verbose',0);
%}

%lstmLayer(outputSize,'OutputMode','sequence')
%lstmLayer(outputSize,'OutputMode','last')
layers = [ sequenceInputLayer(inputSize)
    lstmLayer(outputSize,'OutputMode','last')
    fullyConnectedLayer(numClasses)
    softmaxLayer
    classificationLayer('Name','Pos_Neg_Neu_Classification') ];


            options = trainingOptions('sgdm',...
    'InitialLearnRate',0.05,...
    'Plots','training-progress',...
    'MaxEpochs',10,...
    'Verbose',0);

net = trainNetwork(dataTrain.wordVec,dataTrain.rating,layers,options);

% Use the test dataset to get an accurcy measure
[predictedOutput,~] = classify(net,dataTesting.wordVec);

disp("output size: predictedOutput");
disp(size(predictedOutput));
disp("output size: dataTesting.rating");
disp(size(dataTesting.rating));

accuracy = sum(predictedOutput == dataTesting.rating)/numel(predictedOutput);
disp("accuracy");
disp(accuracy);

return;


% The next three Helper functions are by
% Source Heather Gorr
% https://www.mathworks.com/matlabcentral/fileexchange/68264-classify-sentiment-of-tweets-using-deep-
learning
function dataReadyForLSTM = prepData(emb,documents)
% Prep the data for the LSTM network

    % Transform tweets for model
    dataReadyForLSTM = doc2sequence(emb,documents);

    % Get the max length and left-pad all sequences with zeros
    sizes = cellfun(@(x) size(x,2),dataReadyForLSTM);
    maxLength = max(sizes);
    parfor i = 1:numel(dataReadyForLSTM)
        dataReadyForLSTM{i} = leftPad(dataReadyForLSTM{i},maxLength);
    end
end

% Source see link above by Heather Gorr
function MPadded = leftPad(M,N)
% Add padding to left
    [dimension,sequenceLength] = size(M);
    paddingLength = N - sequenceLength;
    MPadded = [zeros(dimension,paddingLength) M];
end


% I had already downloaded the glove twitter word embeddings and
% placed them in the same directory as the script
% Source see link above by Heather Gorr
function emb = downloadReadWordEmbedding(D)
% Load word embedding
    % Get it from http://nlp.stanford.edu/data/glove.twitter.27B.zip if it isnt
    % found in the workspace or current directories
    fname = "glove.twitter.27B."+D+"d";
    if ~exist('emb','var')                  % check workspace
        if exist(fname + '.mat','file')     % read embedding from .mat file
            c = load(fname + '.mat');
            emb = c.emb;
        elseif exist(fname + '.txt','file') % readWordEmbedding from .txt file
            emb = readWordEmbedding(fname + '.txt');
            save(fname + '.mat','emb');
        else                                % read from web, unzip, readWordEmbedding
            websave('glove.twitter.27B.zip',...
                'http://nlp.stanford.edu/data/glove.twitter.27B.zip');
            unzip('glove.twitter.27B.zip');
            emb = readWordEmbedding(fname + '.txt');
            save(fname + '.mat','emb');
        end
    end
end
```

CA Election Analysis Matlab Code (Requires Matlab’s Text Analytics Toolbox):

electionAnalysis.m

```
disp("Loading GloVe word vectors");
fname = "glove.twitter.27B."+100+"d";
c = load(fname + '.mat');
wordEmbedding = c.emb;


disp("Loading pretrained network");
```

```
load LSTM.mat net;
disp(net.Layers);

%File as in the python project dir move it after data collection
candidatesFile = "/Users/mjscheid/PycharmProjects/twitter_1/Candidates/Candidates.txt";
candidatesData = readtable(candidatesFile,'TextType','string','Delimiter',',','ReadVariableNames',1);
disp("candidates");
disp(candidatesData(:,:));
disp(candidatesData(1,:).screen_name);
canSize = size(candidatesData(:,1));


tweetDir = '/Users/mjscheid/PycharmProjects/twitter_1/rawTwitterData/';

% Add the negative, neutral and positive columns to the table
candidatesData(:,10) = {0};
candidatesData(:,11) = {0};
candidatesData(:,12) = {0};
candidatesData.Properties.VariableNames{'Var10'} = 'negative';
candidatesData.Properties.VariableNames{'Var11'} = 'neutral';
candidatesData.Properties.VariableNames{'Var12'} = 'positive';

disp(candidatesData);


% 1. Loop to get each canidate's screen_name from the table built from candidates.txt
% 2. Build a struct to hold all the tweets from each
%     candidate---tweetStruct.(screenName) -> table of tweets from that candidates
for i=1:canSize(1)
    % Set currentScreenName. It will be used to access the correct tweet
    % table
    currentScreenName = candidatesData(i,:).screen_name;
    disp(strcat("Current candidate's screen_name:", currentScreenName));

    % Open the file contained the tweets
    tweetFile = tweetDir+ strcat(currentScreenName,".txt");

    % Build the current table of tweets and insert it into the struct
    tweetStruct.(currentScreenName) = readtable(tweetFile,'TextType','string','ReadVariableNames',0);
    tweetStruct.(currentScreenName).Properties.VariableNames{'Var1'}='tweet';

    % Perform the same prepocessing that was used on the training data
    tweetStruct.(currentScreenName).tokenizedTweets = preprocessTweets(tweetStruct.(currentScreenName).tweet);
    [~, idx] = removeEmptyDocuments(tweetStruct.(currentScreenName).tokenizedTweets);
    tweetStruct.(currentScreenName)(idx,:) = [];
    tweetStruct.(currentScreenName).wordVec = prepData(wordEmbedding,tweetStruct.
(currentScreenName).tokenizedTweets);

    % display the first tweet, tokenized tweet, and wordvec rep of the
    % tweet
    %disp(strcat("Tweet: ",tweetStruct.(currentScreenName).tweet(1)));
    %disp(tweetStruct.(currentScreenName).tokenizedTweets(1));
    %disp("Word vector rep:");
    %disp(tweetStruct.(currentScreenName).wordVec(1));
    %wv1 = tweetStruct.(currentScreenName).wordVec(1);
    %partialRow = wv1{1,1}(101:105);
    %disp("First 5 values from of the GloVe representation of the second token");
    %disp(string(partialRow));
    % Predict the sentiment of the current candidate's tweets
    [predictedOutput,~] = classify(net,tweetStruct.(currentScreenName).wordVec);
    tweetStruct.(currentScreenName).output = predictedOutput;

    % Show the first tweets sentiment
    %disp(strcat("Predicted sentiment: ", string(tweetStruct.(currentScreenName).output(1))));
    %fprintf("\n\n")


    % sentCounts is an array of the sentiment counts  [negative neutral positive]
    sentCounts = countcats(tweetStruct.(currentScreenName).output);

    % Increment the counts for each sentiment
    candidatesData(i,:).negative = candidatesData(i,:).negative + sentCounts(1);
    candidatesData(i,:).neutral = candidatesData(i,:).neutral + sentCounts(2);
    candidatesData(i,:).positive = candidatesData(i,:).positive + sentCounts(3);


end


%save candidatesData candidatesData
disp(candidatesData)


%save tweetStruct tweetStruct
disp(tweetStruct);


return;


% These helper functions are by
% Source Heather Gorr
% https://www.mathworks.com/matlabcentral/fileexchange/68264-classify-sentiment-of-tweets-using-deep-
learning
function dataReadyForLSTM = prepData(emb,documents)
% Prep the data for the LSTM network

    % Transform tweets for model
    dataReadyForLSTM = doc2sequence(emb,documents);

    % Get the max length and left-pad all sequences with zeros
    sizes = cellfun(@(x) size(x,2),dataReadyForLSTM);
    maxLength = max(sizes);
    parfor i = 1:numel(dataReadyForLSTM)
        dataReadyForLSTM{i} = leftPad(dataReadyForLSTM{i},maxLength);
    end
end

% Source see link above by Heather Gorr
function MPadded = leftPad(M,N)
% Add padding to left
    [dimension,sequenceLength] = size(M);
    paddingLength = N - sequenceLength;
    MPadded = [zeros(dimension,paddingLength) M];
```

```
end
```

Python Code : WriteCandidateDataToFile.py

```
# michael scheid
menu_item = 1
while menu_item != 0:

    name = raw_input("name:")
    screen_name = raw_input("screen_name:")
    party = raw_input("party  r/d:")
    incumbent = raw_input("incumbent y/n: ")
    state = raw_input("state (lowercase):")
    outcome = raw_input("election outcome w/l:")
    percent = raw_input("percent:")
    type_of_office = raw_input("type_of_office (s/l or state/local):")
    name_of_office = raw_input(
        "name_of_office\r\nEx: state_senator\r\n    state_representative\r\n    us_senator\r\n    us_representative\r\n
governor\r\n    lieutenant_governor\r\ninput here:")

    if party == 'r':
        party_bool = 1
    else:
        party_bool = 0

    if incumbent == 'y':
        incumbent_bool = 1
    else:
        incumbent_bool = 0

    if outcome == 'w':
        outcome_bool = 1
    else:
        outcome_bool = 0

    if type_of_office == 's' or type_of_office == 'state':
        type_of_office_bool = 1
    else:
        type_of_office_bool = 0

    # bug when restarting without deleting the file the first new ...must add \n on first write
    # write:name, scree_name, party_bool, incumbent_bool, state
    firstWrite = True
    new_line = name + "," + screen_name + "," + str(party_bool) + "," + str(
        incumbent_bool) + "," + state + "," + str(outcome_bool) + "," + str(percent) + "," + str(
        type_of_office_bool) + "," + name_of_office + "\r\n"

    if firstWrite:
        firstWrite = False
        new_line = "\r\n" + name + "," + screen_name + "," + str(party_bool) + "," + str(
            incumbent_bool) + "," + state + "," + str(outcome_bool) + "," + str(percent) + "," + str(
            type_of_office_bool) + "," + name_of_office + "\r\n"
    with open("Candidates.txt", "a+") as can_file:
        can_file.write(new_line)

    menu_item = raw_input("Menu\r\n1. Continue\r\n2. Quit\r\n")
    if int(menu_item) == 2:
        exit()
    else:
        menu_item = 1
```

WriteTweetsToFile.py

(This file uses a Python interface for the Twitter API. See https://github.com/bear/python-twitter. The interface is covered by the Apache License, Version 2.0)

```
# michael scheid
import twitter_credentials
from TwitterAPI import TwitterAPI, TwitterPager

line_break = "\n"

# twitter_credentials is a separate file not shown because it contains my API keys
api = TwitterAPI(twitter_credentials.CONSUMER_KEY, twitter_credentials.CONSUMER_SECRET,
auth_type='oAuth2')

#open canididates.txt
canididates_file = open('/Users/mjscheid/PycharmProjects/twitter_1/Candidates/Candidates.txt', 'r')

# key -> value pairs.
# screen_names -> real name
name_map = dict({})

var_names = canididates_file.readline()
print("var_names:"+var_names)
for next_line in canididates_file:
    #print(next_line)
    split_line = next_line.split(",")
    name = split_line[0]
    screen_name = split_line[1]
    name_map[screen_name] = name

canididates_file.close()


for key in name_map:
    SCREEN_NAME = key
    REAL_NAME = name_map[key]
    print("Getting tweets from:"+REAL_NAME+"  @"+SCREEN_NAME)

    pager = TwitterPager(api, 'statuses/user_timeline', {'screen_name': SCREEN_NAME, 'count': 1000})

    outfile = open('rawTwitterData/' + SCREEN_NAME+".txt", 'w+')
    count = 0

    for item in pager.get_iterator(wait=3.5):
        if 'text' in item:
            count += 1
            tweet_body = unicode(item['text']).encode('utf-8')
            tweet_body.strip("\t").strip("\n")
            outfile.write('"'+tweet_body+'"'+line_break)
        if count > 1001:
            print("number of tweets:"+str(count))
            break
    outfile.close()


print("Done now!!")
```

Additional helper function’s obtained from the Matlab documentation/ Examples:

preprocessTweets.m:

```
function t = preprocessTweets(tweets,searchTerm)
% PREPROCESSTWEETS  Capture text preprocessing steps in a function
%
% Copyright 2017-2018 The MathWorks, Inc.
% Source https://www.mathworks.com/matlabcentral/fileexchange/68264-classify-sentiment-of-tweets-using-
deep-learning

if nargin < 2
    searchTerm = [];
end

% Preprocess tweets
tweets = lower(tweets);
tweets = eraseURLs(tweets);
tweets = removeHashtags(tweets);
tweets = erasePunctuation(tweets);
t = tokenizedDocument(tweets);

% Edit stop words list to take out words that could carry important meaning
% for sentiment of the tweet
newStopWords = stopWords;
notStopWords = ["are", "aren't", "arent", "can", "can't", "cant", ...
    "cannot", "could", "couldn't", "did", "didn't", "didnt", "do", "does",...
    "doesn't", "doesnt", "don't", "dont", "is", "isn't", "isnt", "no", "not",...
    "was", "wasn't", "wasnt", "with", "without", "won't", "would", "wouldn't"];
newStopWords(ismember(newStopWords,notStopWords)) = [];
t = removeWords(t,newStopWords);

% Remove other common words in tweets
t = removeWords(t,{'rt','retweet','amp','http','https',...
    'stock','stocks','inc'});
t = removeShortWords(t,1);

if ~isempty(searchTerm)
    searchTerm = lower(searchTerm);
    termplural = searchTerm + "s";
    t = removeWords(t,...
        {searchTerm,char(termplural)});
end

end

function tweettext = removeHashtags(tweettext)
torep = {'<[^>]+>', ...                    % HTML tags
    '(?:@[\w_]+)', ...                    % @mentions
    '(?:\#+[\w_]+[\w\''_\-]*[\w_]+)', ...   % #hashtags
    '(?:\$+[\w_]+)',...                    % $tickers
    '\d'};                               % numeric values
tweettext = strip(regexprep(tweettext,torep,''));
end
```

doc2sequence.m:

```
function C = doc2sequence(emb,documents)
% DOC2SEQUENCE Convert text to numeric matrices and prepare for LSTM model
%
% Copyright 2017-2018 The MathWorks, Inc.

% Preallocate
n = numel(documents);
C = cell(n,1);

parfor ii = 1:n
    % Convert text to numeric data for each based on word embedding
    words = string(documents(ii));
    idx = ~ismember(emb,words);
    words(idx) = [];
    C{ii} = word2vec(emb,words)';

end

end
```