Paul Nguyen, Austin Wilson

Chandrasekar Vuppalapati

CMPE 256 Adv Data Mining

18 December 2021

Twitter Sentiment Analysis Project Report

Lord Farquaad’s Star Squad

Table of Contents

[1. Project Description 2](#_Toc90309562)

[2. Project Requirements 2](#_Toc90309563)

[3. KDD 3](#_Toc90309564)

[4. Feature Engineering 4](#_Toc90309565)

[5. High Level Architecture Design 5](#_Toc90309566)

[6. Data Flow/Component Level Design 6](#_Toc90309567)

[7. Sequence/Workflow 6](#_Toc90309568)

[8. Data Science Algorithms 7](#_Toc90309569)

[9. Interfaces 7](#_Toc90309570)

[10. Client-Side Design 7](#_Toc90309571)

[11. Testing 7](#_Toc90309572)

[12. Model Deployment 8](#_Toc90309573)

[13. High Performance Computing 8](#_Toc90309574)

[14. Documentation 8](#_Toc90309575)

[15. Design patterns 8](#_Toc90309576)

[16. Data Engineering 9](#_Toc90309577)

# Project Description

The purpose of this project is to create a web application that analyzes the sentiment of user generated tweets posted to Twitter via Twitter’s developer API. The Kaggle dataset used to train our models consists of four attributes: Tweet ID, Entity (The topic of the tweet), Sentiment (of types positive, negative, neutral, and irrelevant), and the Tweet content itself. In trying to find suitable models, we performed training on the data with a couple neural networks, a recurrent neural network, and a logistic regression model. After running the models and measuring the results of our tests, we exported the RNN model to reload it in the web application. The web application is built using the Flask microservice framework and is designed to retrieve the user’s tweets or post new tweets to their account and display the sentiments of those tweets. As our deployment pipeline, the model is deployed to AWS Elastic Beanstalk via AWS Code Pipeline which handles all of the heavy lifting of setting up and deploying the application.

# Project Requirements

1. UI/Front End
   1. Display a user’s tweets
   2. Allow user to input a tweet and publish it
   3. From that tweet, analyze the sentiment of that tweet and display it
2. ML/Data Mining
   1. ML Model should ingest a twitter dataset for training and validation
   2. Twitter training set will be preprocessed using various methods
   3. ML Model will use NLP (Natural Language Processing) to derive sentiment from data
3. Tech stack
   1. Python as main programming language
   2. Jupyter notebook to perform data mining and building models
   3. Flask as webserver to run trained model
   4. AWS to deploy model

# KDD

Business Understanding:

Goal is to take data from twitter posts and train a model to identify sentiment in order to have it independently identify sentiment from custom tweets.

Data Understanding:

Used twitter dataset from link: <https://www.kaggle.com/jp797498e/twitter-entity-sentiment-analysis>

Dataset contains 4 columns: ID, Topic, Sentiment, Text

Data Preparation:

Reduce the data into only what we need to train a basic model. Disregard tweet ID and topic and filter sentiment to either positive or negative. Convert tweet text into only what is necessary by tokenizing and removing English stop words. Remaining text will contain words that we can derive meaning from.

Data Modeling:

To model the data, we used a basic Keras NLP model. Through Long Short-Term Memory (LSTM) each word remaining in the tweet will add context to the following word to help the model distinguish context and identify the differences between positive and negative tweets.

Evaluation:

Upon splitting the data into training and testing batches, we’ve arrived with an accuracy of 87% with LSTM and at least 85% with various other models. The LSTM model is what we have chosen to deploy for the webapp.

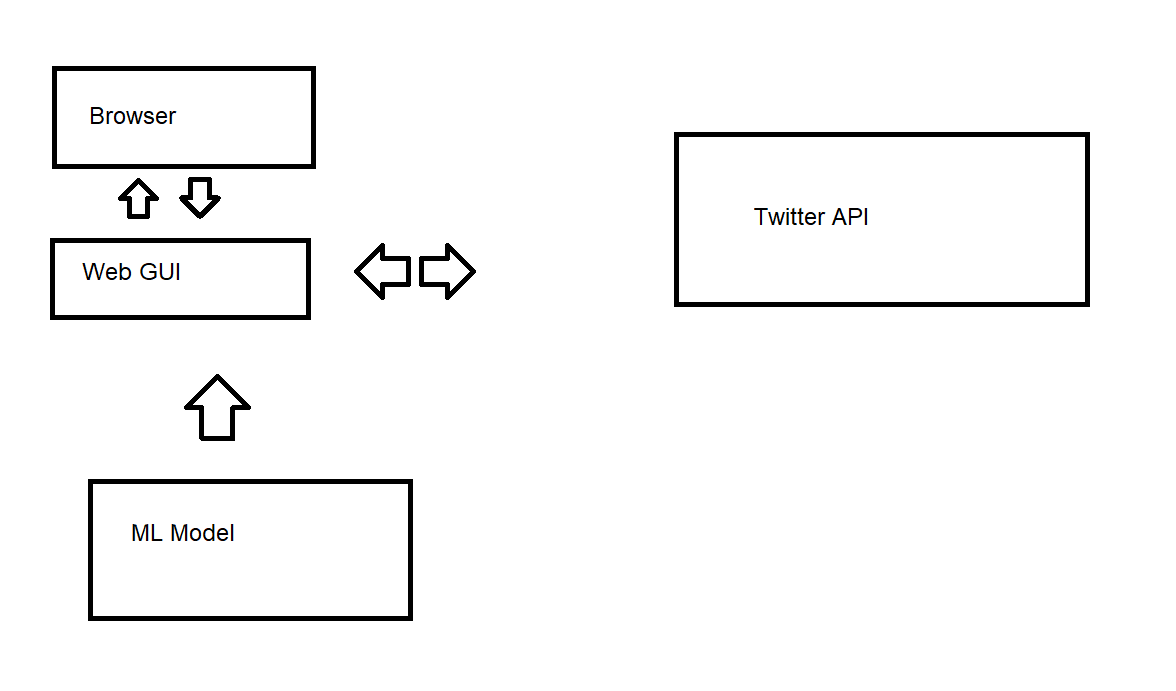
Deployment:

After training the model, we plan to host a basic frontend web app where a user can enter in a tweet and post it to twitter and have it display the sentiment of that tweet after posting. The webapp is deployed on AWS Elastic Beanstalk and upon any changes to the model or switching to a new model, Amazon Code Pipeline will automatically detect any changes to the GitHub repository the code is hosted on to build and deploy the changes.

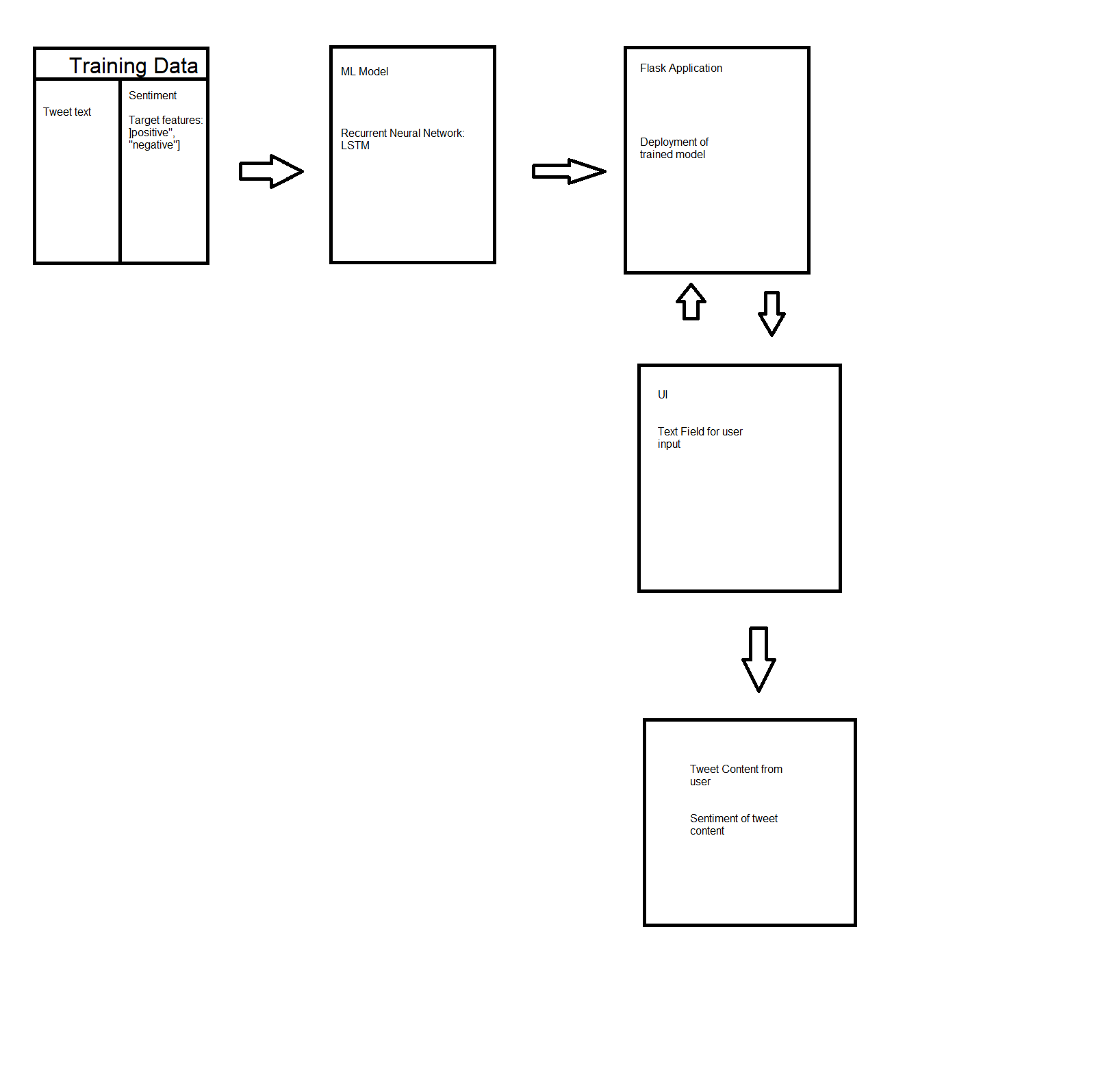
# Feature Engineering

For feature engineering, we are limiting relevant sentiments to either positive or negative for the model to distinguish differences more easily. To process the text, we opted to trim off junk words that provide no meaning or context and lemmatize the remaining words when applicable in order to avoid the model from misinterpreting words with similar meanings but worded differently

# High Level Architecture Design



# Data Flow/Component Level Design



# Sequence/Workflow

As our workflow, we followed the standard CRISP-DM data science workflow.

# Data Science Algorithms

Since we are dealing with context behind text, the algorithm we opted to use some form of logistic regression to classify between two possible values. Our final model is an LSTM implementation of Recurrent Neural Networks since it is our best choice for natural language processing where backpropagation and context are needed.

# Interfaces

All development and testing regarding the model were done in Jupyter notebook. Deployment of the model was handled via webhooks from GitHub to AWS Code Pipeline where it is deployed to AWS Elastic Beanstalk. All monitoring of deployment is managed and displayed by AWS.

# Client-Side Design

For our client-side design, the webapp will display a list of tweets showing the user’s tweets and their color-coded sentiment in either blue for positive or red for negative. Additionally, the user can post a tweet and see the model predict the sentiment afterwards.

# Testing

In testing the model, we split apart our dataset into training and testing batches. After the model runs on the training batch, we run our test batch against the model and record the accuracy by checking the predicted sentiment against the tagged sentiment.

# Model Deployment

Our model is deployed to a webapp hosted on AWS Elastic Beanstalk. Changes to the model or changing the model is automatically performed for us via AWS Code Pipeline so no manual work needs to be done on the host.

# High Performance Computing

The application is hosted on AWS Elastic Beanstalk and performance of the application is automatically adjusted based on the load detected. When necessary, the application can automatically scale upwards and downwards to cut down on costs and save effort in controlling how to scale the application at peak load. During deployment, AWS scaled up our application’s host to a T2.Medium host, accounting for the need to download and build packages needed for it to run and then scaled down to a T2.Nano host as traffic was not utilized and there are no performance consuming tasks on hand.

# Documentation

All documentation is located on the Jupyter Notebook.

Link to source code: <https://github.com/austinwilson1224/twitter-sentiment>

[Live demo](http://twitsenti-env.eba-d3m46afb.us-west-2.elasticbeanstalk.com/)

# Design patterns

The primary pattern used in this project is the regression pattern where we take the features of an entity and match it to a particular classification by finding the differences that make up both classifications.

# Data Engineering

As mentioned earlier in this paper, we’ve applied text transformation onto the dataset’s tweet content column as part of our data engineering process. The goal of this process is to reduce the number of words and features that the model needs to look at to understand the sentiment of the content. By removing stop words and lemmatizing the remaining text, we are able to significantly reduce the amount of words that the model looks at and variations of the same word won’t affect the model’s understanding of their uses.