Software Implementation and Testing Document

For

Group Sentiment Analysis Twitter

Version 3.0

Authors:
Jacob Wharton
Andre Guiraud
Oscar Kosar-Kosarewicz

Programming Languages (5 points)

Python - we are using this language due to its flexibility. It has excellent data science libraries which we need as well as mature web development libraries, allowing us to integrate the components of the project more easily. In addition, all of our group members are experienced in Python.

Platforms, APIs, Databases, and other technologies used (5 points)

- Github for version control.
- Django for web application development
- AWS to acquire training data
- AWS to host the web app
- Tweepy to gather tweets
- PyCLD2 to detect language (instead of Google Translate API via TextBlob)
- TextBlob to parse text and write algorithms for language comprehension
- Schedule to schedule email sender for continuous run
- Bootstrap library for web styling
- numpy and pandas for data processing
- Scikit-learn for machine learning
- chart.js for graph visualization of output data within web browser
- SQLite database
- MySQL database
- AWS RDS to host MySQL database

Execution-based Functional Testing (10 points)

Describe how/if you performed functional testing for your project (i.e., tested for the ***f unctional requirements*** *listed in your RD).*
MachineGym: (Jacob) tested the output of the text analyzers by inputting simple text designed to match one or two hashtags or words in the text, where I already knew what the output should be. I did this repeatedly for various sentiments and for different combinations of words and hashtags.
Tested streamlistener.py by running it several times with different parameters, collecting up to a certain number of tweets and printing them to the console to examine them and make sure the streamlistener was working properly.

Website (Andre): Website was tested running on a local server, along with gmail for sending emails. AWS was tested with increasing versions of the web application.

Machine Learning model (Oscar): Tested the model on a large dataset of tweets to make sure that the model gives predictions with a wide range of inputs.

Execution-based Non-Functional Testing (10 points)

*Describe how/if you performed non-functional testing for your project (i.e., tested for the **non-functional requirements** listed in your RD).*

Website (Andre): Database integrity was tested by attempting to load blank data into the database. It was rejected, maintaining the database integrity. Also, if an email that already existed is entered a duplicate is not created.
Make sure the remote database is secure, as well as the EC2 instance.

Machine Learning model (Oscar): Tested the model on test data so that the model's performance can be quantified and improved.

Non-Execution-based Testing (10 points)

Describe how/if you performed non-execution-based testing (such as code reviews/inspections/walkthroughs).
MachineGym: (Jacob) I always do a code-walkthrough where I re-read everything I wrote, looking for any logic errors, errors in the flow of the program, or any areas where I made a note to go back and change something.

Website (Andre): For the email scheduler a lot of code-walkthrough to ensure that everything was being called where necessary, arguments were being passed correctly, etc.

I tested the HTML code on the browser before running the website too, to try out different bootstrap styling options.

Dealing with AWS errors required a lot of log reading and code walkthrough too.

Machine Learning Model (Oscar): Performed mental walkthrough prior to running code to see whether my expectations match the outcome. Additionally, visual inspect the model's results to the input data so make sure that the results make sense