Software Requirements and Design Document

For

Sentiment Analysis on Twitter

Version 3.0

Authors:
Oscar Kosar-Kosarewicz
Andre Guiraud
Jacob Wharton

1. Overview (5 points)

Our project uses a machine learning algorithm to discover how people feel about certain topics on the internet, or more specifically, hashtags on Twitter. We will use a website where the user can put in the topic they would like to analyze and the year they are interested in. The user will be asked to put in their email so they receive the results when they are finished. For example, the user could search for the topic “impeachment” and get information about the percentage of tweets that are positive about the topic, as well as see the most prevalent emotion such as ‘angry’.

2. Functional Requirements (10 points)

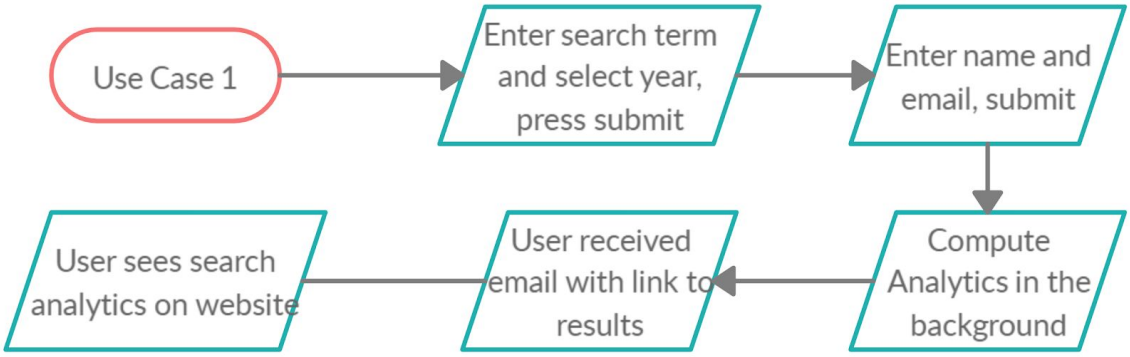
- **DONE:** High priority: Create a way to access tweets for analysis
- **DONE** High priority: Create ML model that is able to analyze tweets and predict their sentiment
- **DONE** High priority: Finish integration of ML model with webapp
- **DONE** Medium priority: Generate a dataset of tweets with more nuanced descriptions than positive or negative so we can create better analytics
- **DONE:** Medium priority: Email is sent as soon as new data is generated for a specific query requested by x user.
    - Note: Instead of being sent as soon as data is generated the app gathers all emails pending and sends them out at the same time.
- **DONE:** Create additional visualizations of our data
- **DONE**: Create additional database tables for our data

3. Non-functional Requirements (10 points)

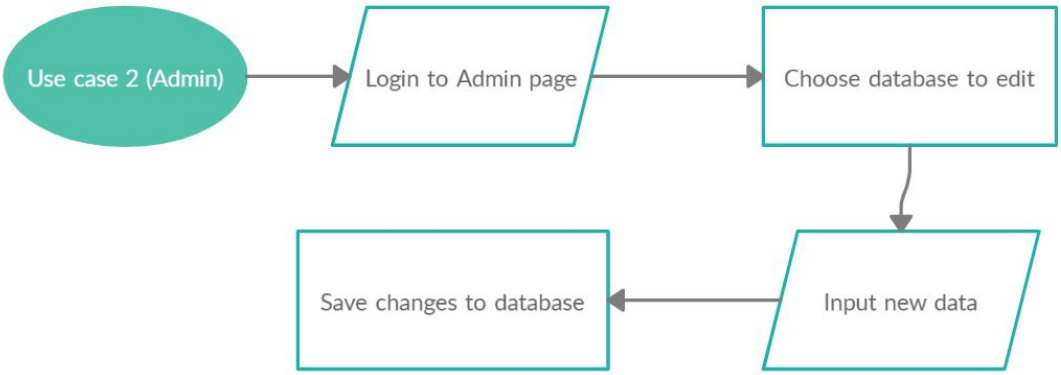
- Have the generated analytics be reasonably accurate
- Test database integrity to avoid errors when querying/saving
- Verify database relations for best performance/quality
- Ensure that the database of tweets is secure from modification outside our functions

4. Use Case Diagram (10 points)

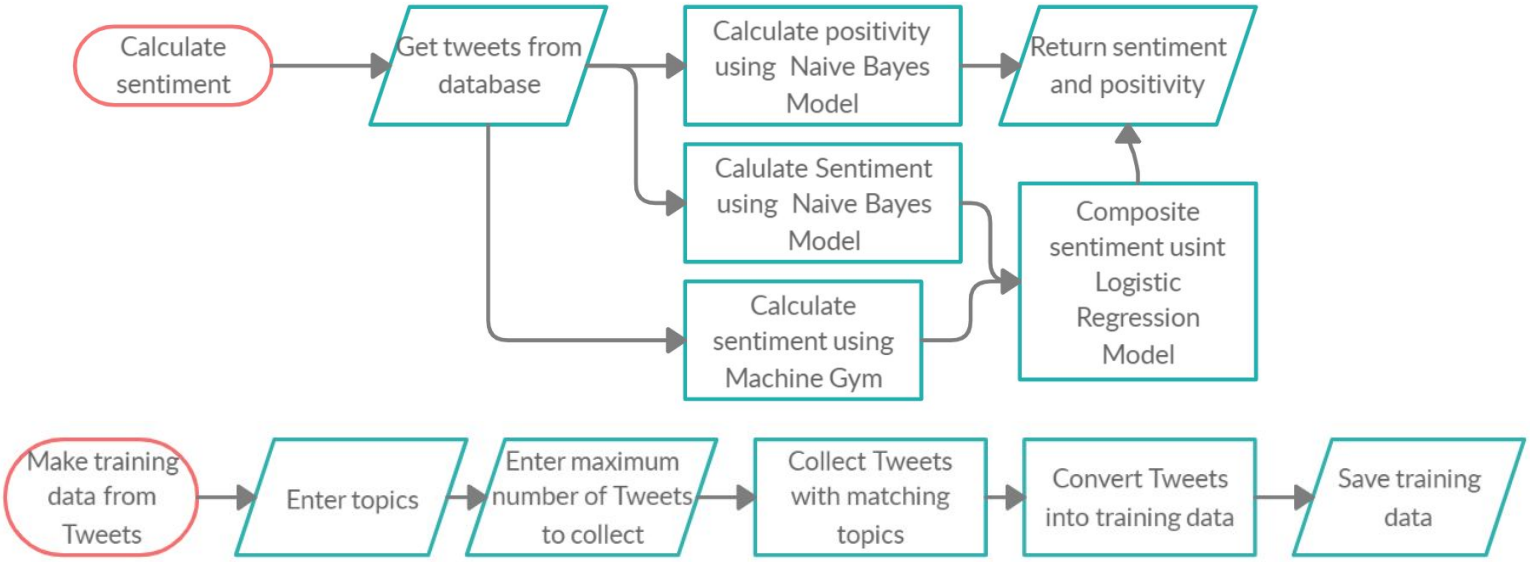
Use case one: this is how the user interacts with the application. The user opens the website, enters a search term into the text box, selects a year from the dropdown and presses submit to start the query. On the next page, they put in their name and email and submit them. One the query is completed, they receive an email with a link to their results. The user clicks the link and is taken to the site where the analytics are displayed.

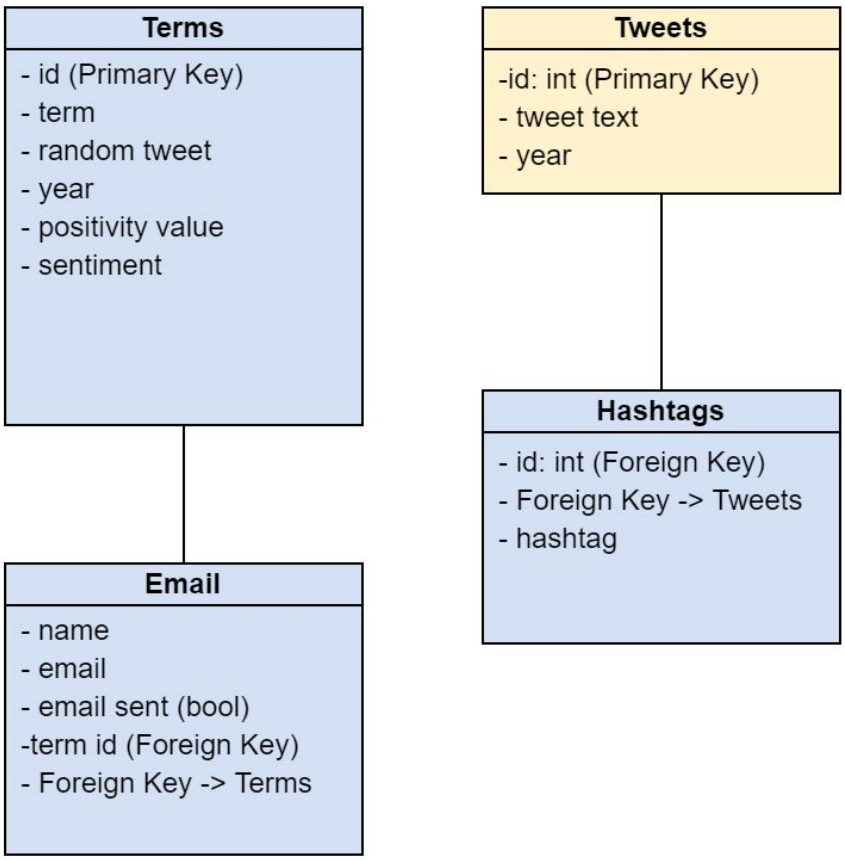


Use case 2 (admin): The admin logs in to the admin page, where he can select the database he wants to edit. Here, he can input new data into the database and save the changes in order to manually update the database.



5. Class Diagram and/or Sequence Diagrams (15 points)





Note: Email is hosted on the EC2 instance where the web application runs. Other tables are on the RDS ‘ebdb’ database.

6. Operating Environment (5 points)

Website can be accessed here: http://twittersa.us-east-1.elasticbeanstalk.com/

The application will run in a (Chromium) web browser, using Python, JS, and HTML. Tested in Chrome.

Necessary for development:
>>>import textblob
>>>import nltk # required by textblob
>>>nltk.download('brown') # required by textblob
>>>nltk.download('punkt') # required by textblob
>>>import tweepy
>>>pip install django

Necessary for streamlistener.py:

- Install pycld2, which is outdated on pip (we DL’d the repo and ran setup.py)

Training model:

- Python 3
- Import pandas, numpy, sklearn

7. Assumptions and Dependencies (5 points)

We can assume that one factor that could affect our project is the company Twitter itself, which owns the rights to their tweets; if they decide not to let us access their API for any reason, that will affect our ability to gather training data, and will ultimately render our application nonfunctional.

At the moment we are limited to a certain amount of API calls per x amount of time, so if too many people make searches at the same time some will never receive an email… :(

Dependencies: Chromium, Twitter API, Google Translate API, textblob, tweepy, django, AWS RDS, AWS ElasticBeanstalk