**Progress Report**

**- Increment 1 -**

**Sentiment Analysis on Twitter**

# Team Members

Jacob Wharton; jtw18f; eyeCube

Andre Guiraud; alg17d; alguiraud

Oscar Kosar-Kosarewicz opk18 Kosaro

1. **Project Title and Description**

Our project is called SentimentAnalysis - Twitter. It will allow users to track the sentiments of tweets posted on Twitter, possibly over time and by location. The sentiments are regarding user-supplied keywords or phrases, the matching tweets for which are mapped to a floating point "quality" value from -1 to 1 using the machine learning model. A graph will display this output in a user-friendly way. We will use the TextBlob Python library to analyze the raw text from the tweets gathered using the StreamListener which accesses the Twitter API.

1. **Accomplishments and overall project status during this increment**

GitHub repo: <https://github.com/eyeCube/SentimentAnalysisTwitter/tree/master/Model%20Training/machinegym>

We have a working machine learning model that can output a 5-star score indicating negative to positive sentiments about the given words / phrases. It uses Amazon reviews as placeholder training data.

There is code in place to download tweets for our use from the Twitter API, and we have signed up for using the Twitter API and been approved for our project by Twitter. We have code as well to output data about the sentiments of a body of text using TextBlob, but we have decided against using this. We have also now secured and downloaded a database of tweets to use for creating the real training data.

We have an idea of what we want the website to look like, but at the moment the layout of the website is purely functional. We have mostly been testing the search function, the email database and navigation, plus stability with querying/saving to the database.

1. **Challenges, changes in the plan and scope of the project and things that went wrong during this increment**

We were considering using TextBlob’s sentiment function to create training data; this uses some black-box algorithm (possibly itself using a machine learning model) to detect the “polarity” and “subjectivity” of a given body of text. We decided against using this because we wanted to go with our own algorithms for creating the training data, since it seemed to defeat the purpose of the project by using an existing model to do this.

What Jacob decided to do is to still use TextBlob, but not its machine learning model. Instead, he will use its text analysis functions to decode English sentences into a rough estimate of its sentiment, using searches for keywords/phrases (taken from a manually collected dictionary). Jacob figured the most intuitive way to do this is to read the text from left to right like a native reader. Qualifiers like adjectives and negators like “not” are read in, and then recorded as a modification to a qualifier variable that affects the quality of any keywords found (for instance, “not happy” is read as a quality of -6 in happiness, the opposite of “happy” by itself, which is a quality of 6 in the happiness scale).

A dictionary of “sad” words/phrases is also used in an attempt to balance the output in cases of a mixture of happy and sad words; the qualities of any matches of opposite-sentiment words result in a loss of a proportionate quality for that sentiment. Then, a sadness scale can be produced by negating the qualities. Other similar context-sensitive analyses will be employed to estimate the sentiments in order to produce large amounts of training data for our model. This naive approach is likely to fail the majority of the time, so a big challenge will be filling out these algorithms to make a more complete sentence analyzer.

Oscar had difficulty training the machine learning model due to his dataset being too large to fit in RAM. Since the model type Oscar wanted to use (Naive Bayes classifier) requires the entire dataset to be loaded into memory, he had to settle for a different model that allowed for partial training with a portion of the data (SGD regressor). This approach works, but not as well as he would like so he will research what other options we have to optimize the prediction model.

We used amazon reviews to train our model due each review having a rating and easy access to the data. However, we have discovered that this may not be ideal as the reviews tend to be lengthy, while tweets tend to be short. This issue will need to be addressed by changing how we approach training the model.

For the website, rather than styling everything ourselves we have opted to utilize Bootstrap and its pre-existing CSS and JS libraries. So, the website currently uses these libraries to present a professional appearance without the need for writing CSS/JS code from scratch.

1. **Team Member Contribution for this increment**

Oscar Kosar-Kosarewicz:

Progress report: IT sections 1 and 2, RD all sections, spoke and edited the video

Source Code: Acquired amazon reviews training data, processed it and used to to train an SGD Regressor prediction model.

Jacob Wharton:

Progress report: contributed to all sections. Setup account with Twitter and have obtained authorization keys for using the Twitter API. Formatting of document.

Requirements/Design doc: contributed to sections 1, 3, 4, 5, 6, and 7. Formatting of document.

Imp./Testing doc: contributed to sections 1, 2, and 5.

Source code: tweepy stream listening; all files within the ~/machinegym/ directory in GitHub repository; implemented using TextBlob, Tweepy, and my prior experience in Python 3.

Video: helped test microphone capabilities and setup video capturing; everything related to tweepy/machinegym

Andre Guiraud:

Progress report: contributed to sections 3, 4 and 6

Requirements and design document: contributed to every section

IT document: contributed to section 2

Source code: everything contained in the website branch, within the project folder ‘SAT\_Website’

Video: all parts pertaining to the website

1. **Plans for the next increment**

Jacob plans to extend the ~/machinegym/machinegym.py functions, as well as improve our collection of sentimental words and phrases, to hopefully create a relatively intelligent English sentence parser to produce the large amount of training data we desire for our machine learning model.

For the website the plan is to get the website to send an email when a variable is changed in the database in order to generate emails for new queries requested by users that were not available to display right away. Additionally, add styling to the website with bootstrap.

Find an alternative to SVB Regression that will give us results that are better suited to our application

1. **Link to video**

Still rendering, will upload momentarily