Progress Report

- Increment 3 -

Sentiment Analysis on Twitter

1) Team Members

Jacob Wharton; jtw18f; eyeCube

Andre Guiraud; alg17d; alguiraud

Oscar Kosar-Kosarewicz opk18 Kosaro

2) Project Title and Description

Our project is called SentimentAnalysis - Twitter. It allows users to track the sentiments of tweets posted on Twitter. The user supplies a topic keyword, which the application uses to find relevant tweets and determine the general sentiment of the tweets. More specifically, the positivity percentage and the most common emotion. A graph will display this output in a user-friendly way. We use the TextBlob Python library to analyze the raw text from the tweets gathered using the StreamListener which accesses the Twitter API; this allows us to generate training data for the machine learning algorithm to improve its accuracy by adding a "human element." Next we use the machine learning algorithm to determine the sentiment. With this we have achieved up to 80% accuracy.

3) Accomplishments and overall project status during this increment

GitHub repo: https://github.com/eyeCube/SentimentAnalysisTwitter/

We have working code that can analyze text such as a tweet and output a sentiment rating for how well it matches a sentiment like anger or happiness. The output can be either "analog," anywhere from -1 to 1, or "digital," being -1, 0, or 1 precisely. This sentiment is score is used as a feature for the sentiment prediction model.

This code now also analyzes other sentiments: bored, fun, fear, and safety. The code has been improved slightly and has been shown to enhance the machine learning algorithm's output when used as an additional feature of the tweets.

We have a working streamlistener which collects tweets live from the Twitter API, and have incorporated that into our web app.

The website was successfully deployed on AWS with expected functionality. We are also able to send emails based on database parameters to alert users of queries they made that have been processed and become available to view. This task is multithreaded to avoid disrupting requests made to the website by users, as it must constantly run to verify the database.

In addition, any time a user searches for a term that is not available the logic for processing this term, collecting a stream of tweets, and running it through our models, is threaded to prevent delays between user requests and webpage return. In effect, we allow the user to put in their email and relax while our server does all the heavy work without taking too much of their time.

We currently have two Models at work:

- We have the positivity model, which uses a Naive Bayes classifier to determine whether a tweet has an overall positive or negative sentiment.
- We also have the nuanced sentiment model, which is composed of three parts:
    - A Naive Bayes model that sorts each tweet into a sentiment category (e.g. peaceful, angry, bored, etc)
    - Machine Gym that analyzes each tweet and calculates the store for each sentiment
    - A logistic regression model that combines the outputs of the two previous models into one final sentiment

We have added a way to measure the performance of a model, allowing us to pinpoint sources of error and improve the model's accuracy.

4) Challenges, changes in the plan and scope of the project and things that went wrong during this increment

MachineGym (Jacob): streamlistener.py was unexpectedly difficult to get working due to problems connecting to the twitter API, along with a host of other issues, but one by one I worked through them. One involved our choice of language detection algorithm for filtering out English tweets from all the other languages (so many Portuguese); Google Translate's API had a max number of requests it could process in some given length of time. So, I was not able to use that to quickly collect training data for machinegym. I managed to find a compact, yet efficient language detection library for python, called pycld2, which runs the language detection locally on the server instead of on Google's servers. It is also much more lightweight which means it's less accurate. But, for our purposes, I think this is sufficient and actually ideal, given how lightweight it is.

I ended up not adding emoticons to machinegym's sentiment parser. I never once saw a tweet with an emoticon in it. Not sure if those are still a thing.

ML Model (Oscar): We have replaced the SGD Regressor with a Naive Bayes which have greatly improved results. We have also used Stanford's Sentiment140 dataset for some of our training data which proved to be a better fit than the Amazon Reviews.

<u>Website (Andre)</u>
For this increment, again, getting AWS to play ball was pain. As we kept adding libraries to our project each time I uploaded a new version of the app I had to verify the logs that elastic bean provides because something would break. A list of thing includes: having to modify wsgi file after importing numpy and pandas, updating SQLite myself because of an incompatibility, installing things with pip because for some reason they failed when installing from the requirements, verifying the processes with top on the server, modifying the wsgi file again… because, and etc.

Aside from the issues with AW, I had issues with the increasing difficulty of execution-based testing as the complexity of the application and functionality grew. When SQL queries were handled in the server the connections would timeout because of the time it took for the processing/analyzing of information, which was fixed by having the connection timeout value modified

5) Team Member Contribution for this increment

Oscar Kosar-Kosarewicz:
Progress report: Edited RD, IT and progress report, edited video and implemented the Naive Bayes sentiment prediction models

Jacob Wharton:
Progress report: contributed to sections 2, 3, 4, 5, 6. Worked on expanding machineGym.py and streamlistener.py (formerly sentiment.py)
Requirements/Design doc: contributed to sections 6
IT document: contributed to sections 2, 3, 5
Source code: all files within the ~/machinegym/ directory in GitHub repository (new files: fear.py, safe.py, bored.py, fun.py, streamlistener.py) including documentation. Javascript: chart.js and json packing/unpacking.
Video: everything relating to machinegym.py / streamlistener.py (script, recorded video/audio files)

Andre Guiraud:
Progress report: contributed to sections 3, 4 and 6
Requirements and design document: contributed to sections 2, 5, 6 and 7
IT document: contributed to sections 2, 3, 4 and 5
Source code: most things contained in the website branch, within the project folder 'SAT_Website'
Video: all parts pertaining to the website

6) Plans for the next increment

Jacob

If we were to continue this project in the future, I would like to see the machine learning sentiment analysis continue to improve more and more as its growth potential is unlimited. That MachineGym works at all is satisfying to me, but it would be interesting to see how improving it would affect the model's ability to pick out specific sentiments with more confidence and accuracy.

Andre

I would continue working on the website and fixing a couple of quirks that slipped through. I would also like to make it a bit more stylish, display more data for a term and different graphs, add an option to search a range of years to see the mood swings over time, and most importantly: optimize functionality.

Oscar

If I would continue working on this project, I would continue to optimize and extend the functionality of the prediction models so that they can be more accurate and provide a more descriptive analysis of the data. I would accomplish this by engineering a more robust training dataset from which we can create more robust models.

7. Video

https://youtu.be/8ejwPnRiZaQ