Pulse: Reddit Sentiment Analysis

Monday, April 15, 2019

Trevor DeBardeleben (tdebar2@lsu.edu)

Cade Thomasson (ctho253@lsu.edu)

Taylor Smith (tsmi235@lsu.edu)

<u>Introduction</u>

Motivation:

For a variety of reasons, companies need to collect data to determine how individuals are discussing their brand. Our original motivation for this project was to create a tool companies can use to determine the sentiment of Reddit comments mentioning their brand. One use for an application such as this would be to determine how people are talking about your brand before and after an advertising campaign to determine how successful the campaign was in increasing brand sentiment. As we developed our project, however, our motivations grew. We realized a product like this would also be useful for political candidates to determine the sentiment of comments mentioning a particular candidate.

Project Description:

For our project demo, we focused on the ladder motivation. This is simply because it would be hard for us to predict in advance when a company will roll out a new advertising campaign or launch a product to collect a significant amount of data before and after the launch. Sentiment of political candidates, however, is always changing. It is interesting to see how particular subreddits feel about certain candidates, and it will be interesting to see how that changes as the campaigns evolve.

In our project we monitor two subreddits: r/politicaldiscussion and r/politics. We monitor all comments posted on each subreddit, and if a post mentions a major candidate in the 2020 primary, that comment is stored in an HBase table and is run through sentiment analysis. We calculate the average sentiment for each candidate every hour, and that data can be viewed on a web frontend.

Division of Roles:

Cade created the server cluster and configured all services. Created initial code for redditIngest.py and submitToHbase.py. He set up the NaïveBayes model using MLlib and performed the sentiment analysis of comments using that.

Taylor focused on storing comments to HBase by modifying redditIngest.py to parse Reddit comments and modifying submitToHbase to pull from the Kafka stream and store it into HBase. Enhanced code to calculate hourly sentiment averages. Created script to submit hourly averages to the averages table in HBase.

Trevor focused on storing comments to HBase by modifying redditIngest.py to parse Reddit comments and modifying submitToHbase to pull from the Kafka stream and store it into HBase. Created the initial code to calculate hourly averages of sentiment for each candidate. Developed web application, including REST API and React frontend located in the "web-dev" folder.

<u>System Design:</u>

Chosen Big Data Frameworks:

Apache Spark, Kafka, HBase, MLlib

Chosen Datasets:

We collect the data ourselves using the Reddit API and The Python Reddit API Wrapper (PRAW).

Detailed Description of Components

1. Data Collection *(redditIngest.py)*
    a. Our data collection component streams a comments posted to a subreddit from a specified list of subreddits using PRAW. If the comment contains a word in the specified list of keywords (in our case political candidate names), a sentiment analysis will be run on the entire comment using vaderSentiment and sent via a Kafka topic to a spark job which also runs sentiment analysis on the comment using MLlib. After the analysis is complete, the comment along with other useful data such as the user who posted the comment, the subreddit, the sentiment rating, and the timestamp are posted to a Kafka topic for the next phase of processing.
2. Data Storage *(submitToHbase.py)*
    a. submitToHbase.py
        i. In our submitToHbase.py script, we create a Kafka Consumer to listen for anything posted to our Kafka topic. When redditIngest.py or the MLlib spark job posts their data to the topic, submitToHbase.py, or it's MLlib counterpart, will retrieve that data via this consumer. Using the python HappyBase library, submitToHbase.py takes the data retrieved from the consumer, formats the data, and adds to an HBase table.
    b. HBase table structure
        i. In HBase, we have a table for each processing method, Vader and MLlib, for each candidate (ex. mllib_sanders). Each of these tables has one column family: "comments." The row id for each comment is "<timestamp>--<user id>". We chose to store each candidate in a separate table for faster data retrieval. Another option we considered was just to have a table for each subreddit that we are listening to. This would have been useful if we were interested in the overall sentiment of a subreddit. Because we are interested in the sentiment of comments about a particular candidate, storing comments that mention that candidate is the best way of storing for efficient data retrieval.
3. Sentiment Analysis
    a. vaderSentiment
        i. vaderSentiment is a Python library based off of research done by C.J. Hutto and Eric Gilbert on sentiment analysis of social media text. The library "implements the grammatical and syntactical rules described in the paper, incorporating empirically derived quantifications for the impact of each rule on the perceived intensity of sentiment in sentence-level text" (https://github.com/cjhutto/vaderSentiment). The library allows you to input text to vaderSentiment.py, and the library will calculate the sentiment of that input. The library will output ratings for positive, negative, and neutral sentiment as well as the composite score. In our project, we only use the composite score.

    b. Spark's MLlib (Machine Learning Library)
        i. Spark's MLlib was designed to parallelize machine learning utilizing the features spark provides. We first trained a naive bayes model using a set of 1.6 million categorized tweets (https://www.kaggle.com/kazanova/sentiment140). We stripped out unneeded common words such as a, the, and and and any emoticons or websites to sterilize the data set. This model was then saved and used in the main Spark job to calculate the polarity of a given sentence. This polarity was normalized to match the range of scores received from vader to ease data analysis and graphing.

4. Data Processing
    a. averageSentiment.py
        i. In order to have hourly averages for use in our frontend React charts, we created a new hbase table called vader_averages to store the averages for the vader calculated sentiment. This file scans the target hbase tables and calculates the hourly average sentiment for each keyword and inputs them into the vader_averages table. These hbase entries are then used in the getHourlySentiment file below
    b. getHourlySentiment.py
        i. This file is a helper file for the React framework to query the vader_averages table and pull the hourly sentiments that are needed for the chart. It takes the sentiment from each hour and outputs a json file that is used to display the graph on the frontend chart.
    c. averageRollingSentiment.py
        i. This file utilized Python's Data Analysis Library, Panda, to calculate a rolling average of the sentiments. These are useful to smooth out datasets in order to recognize larger scale trends.

5. Web Dev
    a. Backend
        i. For our data visualization, we needed a simple way to query our database for information needed on the frontend, so we built a simple REST API using Flask. The only route used is /averages, which takes 3 query parameters: table name, start time, and end time. When this route gets called, it queries the HBase database for the specified information, and it returns that data in JSON format.
    b. Frontend
        i. The frontend of our application is built using React. React allows for rapid prototyping and has many libraries useful to our project. For data retrieval from the API, we use Axios. This is a data fetching library that makes retrieving data from our API simple.
        ii. For our date range picker, we use a library called BluePrint.js. Using this allows users of the site to use an easy to way of visually selecting a date range for data retrieval.
        iii. For the data visualization, we use a Chart.js React wrapper called "react-chartjs-2". Our API is built to respond with data in JSON format that is

easily interpreted by this library so that there is very little data processing that has to be done on the client side.

iv. Upon loading our frontend, a user is greeted with a date range picker and a list of candidates we have tracked sentiment of. A user can use the default date range, or modify it to use the date range of their interest. Each candidate has a checkbox by its name. When the box is checked, data for that candidate is retrieved from the API gets appended to the chart.

Evaluation & Test

Software Manual:

You must have all big data frameworks used and their dependencies installed onto your machine.

How to Run Main System:

note: using a window multiplexer such as tmux will make running our project easier, as you can create multiple windows and panes to run our files all in one single view

1. Start retrieving reddit comments by typing the command “python redditIngest.py”
2. Start the MLlib Spark job by running “spark-submit --class "NaiveBayesAnalysis" --master yarn --executor-memory 16G naive-bayes-analysis_2.11-1.0.jar”
3. Begin storing comments in the database by typing the command “python submitToHbase.py”
4. Begin storing mllib scores by running “mllibSubmitToHbase.py”
5. Start the backend by running the command “flask run --host=0.0.0.0” inside the backend folder.
6. Frontend
    a. navigate to the frontend folder and type the command “npm install” to install all dependencies
    b. type the command “npm run start” to load up the frontend, which will be viewable on port 3000

Other Scripts:

1. Running averageSentiment.py with the command “python averageSentiment.py” will compute the hourly averages of all comments leading up to the time you run the script, and it will store those averages in the appropriate HBase table.
2. Running averageMllibSentiment.py will compute the average sentiment for the previous hour. This and “averageSentiment.py” are placed into a cron job.

How to Use:

Viewable at: http://dae12506.ngrok.io/

1. Select a date and time range in which you would like to view sentiment

2. Each candidate will have a checkbox next to it. Check the box of each candidate whose average sentiment you would like to view, and that data will be displayed to the graph below
3. You can uncheck a box to remove that candidate from the graph.

   note: You may notice MLlib rolling averages does not honor the date range you select. This is because we added calculations for rolling averages after completing the project in order to see if we could find any interesting insights, but we did not have time to fix this bug where it works improperly with the date range.

Test Environment:

The test/runtime environment is a Hadoop/Spark ecosystem set up by cloudera manager on two Dell servers, one being the namenode and master, the other simply running as a datanode. Cloudera manager was used to install and manage Hbase, HDFS, Kafka, Spark, Yarn, and Zookeeper as well as server performance monitoring and reporting. The two servers are connected across a gigabit switch on a fiber network.

Test Cases:

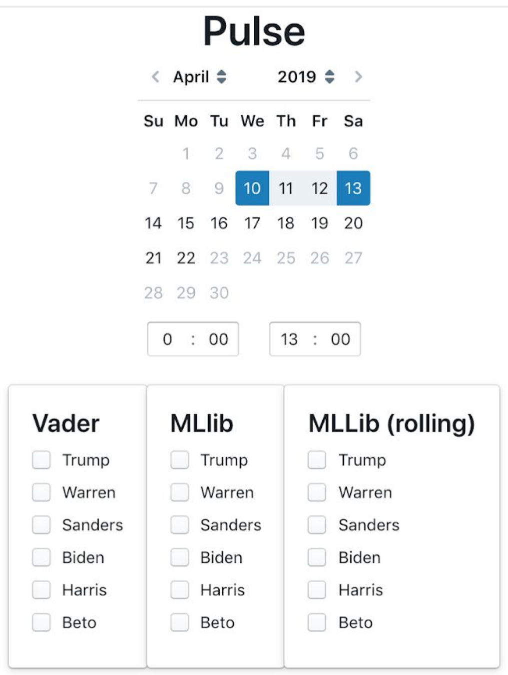
We did not write any explicit testing such as unit tests or integration tests. The scope of this system was small enough that we felt comfortable iterating our code and manually testing changes. However, in future iterations of this program, it would be worthwhile to set up a testing suite to make iterating easier.

Because our project is fairly modular, it was easy to test that each component works during development. For example, in redditIngest.py, we can view in the console all comments that it receives to confirm that it is in fact reading comments from reddit. We also would post comments on subreddits, just to make sure the program would detect that comment was posted. In submitToHbase, we could look at the HBase tables and ensure they were updated with new content. When developing the backend, we were able to hand check a few examples just to make sure that calculations (such as calculating average) were outputted correctly. And changes to the frontend were easy to test and ensure they work properly. Again, this is not an ideal way of testing, but due to the time constraint, we found it most convenient.

Conclusion

In conclusion there is much more that could be done with this project. It has been shown that this is viable, as even the non-parallelized vader library scripts had no trouble keeping up with the influx of reddit comments. When the system was being built out we were storing comments in order to gather a data set. Once we had the sentiment analysis set up it was run retroactively on the past data gathered. This was the largest influx of data at one moment for the system. Our Spark job combined with MLlib was able to clean, analyze, and store 25,000 comment chunks of data in about 10 seconds each. This is far greater than any amount of comments we received at any one time from streaming the comments real time. This shows that processing speed is not a limiting factor in scaling this project.

Appendix



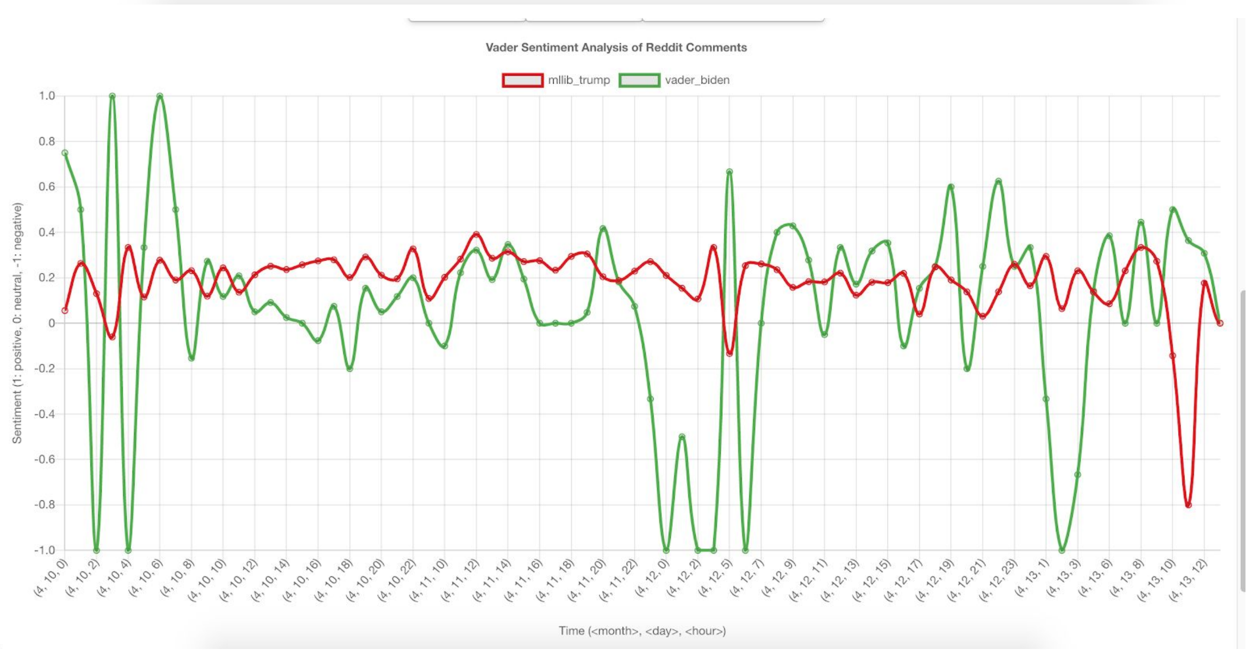
Pulse

April 2019

Su	Mo	Tu	We	Th	Fr	Sa
	1	2	3	4	5	6
7	8	9	10	11	12	13
14	15	16	17	18	19	20
21	22	23	24	25	26	27
28	29	30				

0 : 00   13 : 00

Vader	MLlib	MLLib (rolling)
☐ Trump	☐ Trump	☐ Trump
☐ Warren	☐ Warren	☐ Warren
☐ Sanders	☐ Sanders	☐ Sanders
☐ Biden	☐ Biden	☐ Biden
☐ Harris	☐ Harris	☐ Harris
☐ Beto	☐ Beto	☐ Beto



File Name	Number Of Lines
redditIngest.py	56
sumbitToHbase.py	82
averageSentiment.py	53
getHourlySentiment.py	27
SentimentChart.js (web-dev/frontend/src/components/)	195
api.py (web-dev/backend/)	32
averageMllibSentiment.py	56
mllibSubmitToHbase.py	82
NaiveBayesAnalysis.scala	105
rollingAverageMllib.py	57

All files can be downloaded at:
https://lsumail2-my.sharepoint.com/personal/ctho253_lsu_edu/Documents/Forms/All.aspx?cid=f4b9653c-7197-4321-844e-bba51bc1d3ce&FolderCTID=0x0120003C02D65D6F1121449AE5A8E4F620D1AF&id=%2Fpersonal%2Fctho253_lsu_edu%2FDocuments%2FPulse%2Fweb-dev%2Ffrontend%2Fsrc

Citation

VaderSentiment: lexicon and rule-based sentiment analysis

- Hutto, C.J. & Gilbert, E.E. (2014). VADER: A Parsimonious Rule-based Model for Sentiment Analysis of Social Media Text. Eighth International Conference on Weblogs and Social Media (ICWSM-14). Ann Arbor, MI, June 2014.
- https://github.com/cjhutto/vaderSentiment

Blueprint: React UI Toolkit

- https://blueprintjs.com/
- Used for styling UI components on the Frontend

react-chartjs-2: A React wrapper for Chart.js 2

- https://github.com/jerairrest/react-chartjs-2
- Used to generate charts to visualize data

Spark MLlib setup guide

- https://github.com/P7h/Spark-MLlib-Twitter-Sentiment-Analysis/wiki/Blog
- Used to set up training the naive bayes model and as a guide for working with MLlib