Weather and Sentiment

Robert Huselid, Pranab Islam, Akash Mukkavilli, Khalil Sayid

Abstract

We hypothesize that there is a positive, measurable relationship between good weather and sentiment. To that end, our group has to download, filter, join, process, merge, and finally, analyze a very large source of data (4 TB+). Specifically, we use data from NOAA weather stations (multiple GBs) and twitter (4TBs) to do a two stage analysis. We use this analytical framework to study both the primary relationship (that temperature and sentiment are related) as well as a series of related topics such as time series sentiment analysis, clusters of sentiment, as well as user to user similarity comparison.

Data

Data was scraped from archive.org (scraped to download files, not from the HTML) and weather data was requested from the NOAA. The twitter data was downloaded in a series of tar files with a bash script, then unzipped, concatenated, and processed to create a filtered tweet json. The weather data had to be split apart, reemerged with itself, and then merged onto the tweets. The datasets were merged with MapReduce and then made into a new file. This was done by matching tweets with the closest weather station based on euclidean distance of latitudes and longitudes. This is roughly a nearest neighbor approach, but unlike running this on a single machine, this merge does not necessarily lead to tweets merging with their closest weather station, We were willing to accept that compromise because weather is not localized (i.e. the weather in Evanston is roughly the same as the weather in Hyde Park). Therefore, we were willing to accept some accuracy decrease for a large efficiency increase (this is important since we are using a n*m complex algorithm merge).

Methodology:

Temperature Data and Station Date Merge

The temperature data was obtained through the National Center of Environmental Information, which is part of NOAA. To narrow down the scale of our project, we utilized the daily temperature data for 2018 only, which was a 1.12 GB file. It was problematic that the dataset only contained the station id, the date, the type of weather (Precipitation, Snow, Minimum Temperature, Maximum Temperature etc.), the quantity of that weather, and the location. This file was quite large and we attempted to create the dataset as a SQLITE Table and a MYSQL table. It was apparent however, that both those programs were unable to support such a large dataset. Thus, we ended up using a Pandas dataframe to obtain the dataset. The temperature dataset only contained “MAX_TEMP” and “MIN_TEMP”, so within pandas, we took the average between the maximum temperature and the minimum temperature to create our temperature value. As stated above, the temperature dataset only contained the station id, which made comparing the locations of each tweet to each station very difficult. Luckily, there was a dataset that contained all the latitudes and longitudes for the weather stations. Narrowing down the geography to the continental United States, we used a Pandas *left merge* to combine the two datasets. The final outcome of the dataframe is below:

stationid	date	average_temp	latitude	longitude	location
AR000087418	20180101	24.90	32.8330	-68.7830	MENDOZA AERO
AR000087418	20180102	25.20	32.8330	-68.7830	MENDOZA AERO
AR000087418	20180103	20.70	32.8330	-68.7830	MENDOZA AERO
AR000087418	20180105	29.40	32.8330	-68.7830	MENDOZA AERO
AR000087418	20180106	29.50	32.8330	-68.7830	MENDOZA AERO
AR000087418	20180107	17.40	32.8330	-68.7830	MENDOZA AERO
AR000087418	20180108	27.75	32.8330	-68.7830	MENDOZA AERO
AR000087418	20180109	26.10	32.8330	-68.7830	MENDOZA AERO

After a brief communication with one of NOAA's meteorologists, William Brown, we were able to confirm that the temperature is being measured in degrees celsius.

Tweet Sentiment Analysis

Sentiment analysis was done using NLTK. Specifically, emojis and stop words were removed and/or ignored in the resulting strings that were analyzed. Removing stop words has a complicated impact on efficiency since it forces you to loop over the text an extra time to remove the words, but it also means there are less words in the final string that is put through NLTK. In my experiments on the matter, it seems that the removing stop words increases run time slightly, but this could be a peculiarity of the test files our group used. We used a single float a measure of sentiment (the compound sentiment polarity), instead of doing some multidimensional analysis in order to make the measure easy to put into a regression and derive insights from. The code for this aspect of the assignment was written twice: one using mapreduce and the other using regular IO in python (both versions work).

Map Reduce Tweet to Weather Station

The twitter data and the weather station data both contained the longitude and latitude necessary to associate each tweet in a given day to a weather station on the same day. This would allow us to compare the temperature of that day with each tweet's sentiment in our regression analysis. The basis of the MapReduce was to make the keys the date and the values all the information associated with each tweet (latitude, longitude, sentiment, etc.). We also passed along the temperature data to reference in the reducer. A nested loop was utilized to compare the distances of all the stations to the distances of all the tweets. We used the euclidean distance formula comparing the latitudes and the longitudes. After checking each

station against each tweet, the temperature of the station with the the shortest distance was yielded with the sentiment associated with the tweet. This was written into a text file to be used in the regression analysis.

Caveat: Sampling Data

The size of the dataset for twitter was too large to be hosted on a Google VM so we were forced to conduct this analysis on only the month of October. Thus, we cut down the temperature data and the twitter data prior to the MapReduce. The MapReduce was $O(n, m) = n \times m$ complexity, which required us to use a sample size of 500 tweets to improve run time and test results.

Regression analysis:

The results from our regression analysis show that there is little in the way of a statistical relationship between temperature and sentiment in the samples we tested. If there were more time in the project, perhaps a relationship would arise if the regression model were able to be better fine tuned to the data and larger samples of results could have been tested. Regression was implemented once with mapreduce and numpy and again with mapreduce and just base python. This was a standard direct computation of regression approach wherein matrixes are computed, inverted, and multiplied such that the least square optimal coefficients were chosen.

Clustering Analysis:

We sought to determine whether there exist certain regions of the US that have remarkably high sentiment. We approached this problem using Map-Reduce, aggregating the sentiment values from each user, grouping them into bins, and averaging the sentiments for all users in each bin. Later, we determined that a Top-N approach to the problem would be better since we would have to parse the entire output file to find the areas with highest sentiment. We observed that

the top 5 cities in our sample with the greatest sentiment were Stafford, CT, Los Angeles, CA, Tappahannock, VA, Donovan, IL, and Alamogordo, NM (see Table 1). We converted regions into cities by using an online reverse geo-coding tool. To ensure our Mr-Job code was functioning correctly, we also iterated through the sample data directly on our local machine. The results appear to match each other. We attempted to map our rectangular partitions onto a Matplotlib figure, but handling two-dimensional objects and shading them appropriately (a gradient where darker colors indicate higher sentiment) proved to be difficult. Given more time though, this could be completed.

	Average Sentiment	Number of Tweets
(706, 94)	0.195755	2014
(93, 54)	0.277538	704
(517, 101)	0.235872	385
(562, 52)	0.254299	370
(664, 83)	0.209450	318

Table 1: Sentiment of Top 5 regions in US (using October 2018 Twitter Data)
The first column identifies the region. Each region is roughly 46 square miles in area.

Related Topics and Findings:

While in the process of finding out if there existed a relationship between weather and tweet sentiment, we wanted to see if we could in some way control for general moves in tweet sentiment that may be due to masses of people in the same region or in the United States as a whole reacting to or motivated by some other external localized event. This led us to constructing a time series of each users' tweets sentiment over the time period of our tweet dataset. A lot of interesting analyses could be done with this data.

There were a lot of difficulties with the theory that supported that this may be a way to control for random anomalies that were not weather related because our dataset eventually had to be filtered into one month of tweets October 1, 2018 - September 1, 2018. Originally it was supposed to be a year's worth of tweets. Also, one critical insight is that no two users tweet at the same times. In fact, it was also relatively hard to find users who tweeted in the same day for many days in a dataset of only a month. Users may tweet three out of thirty one days. And on top of that, there are certain events that may happen at around midnight and cause two users to tweet but the resulting tweets may have been classified on two different days even if they were minutes apart. There were a number of workarounds we had for this such as making time series comparisons based on tweets that were separated by some delta t unit of time (potentially 3600 seconds for 1 hour).

The analysis was getting quite statistically heavy and it felt relatively disingenuous of an analysis. We decided that in order to stick with the spirit of the class and compute interesting findings that deal with large computations, we would simply build an index of national average tweet sentiment. Choosing random users for each day in our data set, we aggregated and averaged the sentiments to serve as an index. This is very analogous to the S&P 500. We then compared the time series of each user's tweet sentiments to the index to obtain a "beta value" which represents how volatile a person's tweet sentiment is to the national average. Of course each user still did not tweet every single day in the data's time frame so even this measure is somewhat questionable but if we assume that each day's tweet sentiment on aggregate is mostly independent from the previous days, this analysis is sensible.

We then, through regular python file handling, computed each user's beta values and stored that result into a file. In that file, user_final.json, each user_id is mapped to their beta value and a list of latitudes and longitudes of their tweets. We then compared every user to every single other user in this file through map reduce to see which users had similar beta values. If a user's beta values was within 0.05 of another user's (including their own), the map reduce outputs the user_id of the original user followed by a list of similar sentiment volatility user_ids and the latitudes and longitudes of those similar user's tweets.

Comparing each user to each other user resulted in a computation that had to compare roughly 27,000 final users to every other user among the 27,000. This is a considerable calculation that certainly needed us to run map reduce on a cluster. If our dataset had been twice as long, this calculation would crudely be multiplied by 4x. If we had a years worth of data, then this would crudely increase the calculation time by 144x.

Due to the fact that we had to submit our project early because of graduating seniors in our group, we could not perform certain analyses on our final map reduce product cluster_comparison_final2.txt (this may likely be stored in google drive since it's too large for github but we will have a file that directs you on how to retrieve it) We could have, with a couple days more time, been able to derive some results as to which users in which locations generally tweet with the same sentiment volatility. We could identify groups of users with similar sentiment volatilities and map them geographically using their coordinates (see figure below). We could see which pockets of America have strong sentiment tweets. We could have located the nearest 10 similar sentiment volatility users to every other user and derive some analysis from that. We can see which states represent the nation's average "feeling" as well. We could identify groups

of similar sentiment volatility users and look more into their twitter profiles to see if these users are part of the same cultural sub groups. We could also see if similar sentiment volatility users share certain traits such as: number of followers or number of tweets.

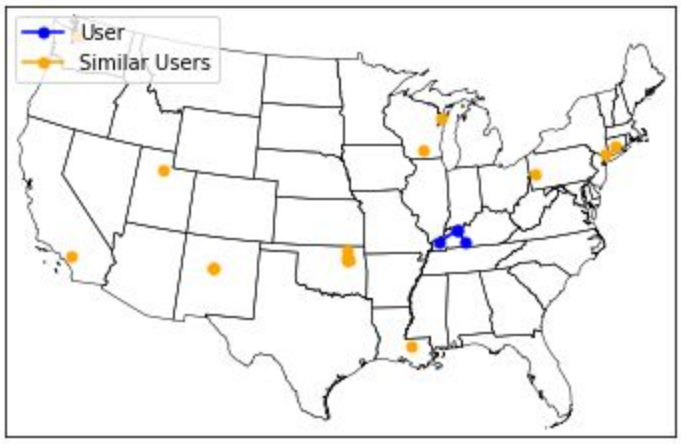

Figure 1: [Geographic plot of a User's tweets against users with similar sentiment]

Regardless, we have built a framework to analyze tweets compared to a national average and compare every single geo-tagged user to each other. We can show each user the graphed locations of each of the people in their personal similarity lists. It makes for a pleasing graph and is rewarding to see who is similar to whom in America along the tweet sentiment dimension.

A lot of this analysis did not deal with "big data" until we reach the comparing each user to each other user step. If there were 100,000 users that were eventually filtered out and deemed valid, a single machine is more than capable to iterate through each of those users and calculate sentiment scores and what not so a lot of the code for this was done without MRjob in time_series.py. But, comparing 100,000 users to 100,000 users becomes a cumbersome task for 1 machine since storage and speed might finally become an issue. MR_Compare_Users.py does this user to user comparison. Even though not necessary for our particular set of data since it ended up being filtered very well and the dates were shortened to only 1 month due to

data storage and retrieval issues, we also built MR_time_series.py that takes in a file of geo-tagged tweet dictionaries and outputs the structures necessary to do locational analysis for each user and create sentiment time series for them. Theoretically, the file could have had 10 times the data than the dataset we had to settle one due to time and data restrictions.