Anna Mendelson, Elizabeth Nammour, and Ali Kozlu
NETS 150
Due May 4, 2015

Chillier, Not Chiller: Analyzing Twitter Data to Confirm the West-Coast-Is-More-Chill Hypothesis

Introduction

West Coasters love to brag about their laid-back demeanor while stereotyping East Coasters as neurotic cynics who cannot distinguish the smell of a rose. But are these claims justified? We sought to evaluate the west-coast-is-more-chill hypothesis by analyzing the words used in tweets sent from both sides of the country. By comparing the frequency words that we deemed suggestive of "chill" personalities (e.g., "chill," "cool," "nice") to the frequency of those that suggest neuroticism (e.g., "omg," "ugh," 'hate," "wow"), we calculated the "chill/neurotic ratios" of various locations. Based on these metrics, we evaluated "chillness" on both coasts in attempt to settle the civil dispute of our generation.

Hypothesis

We examined the following hypothesis: the ratio of "chill" words to "neurotic" words used by the West Coast is greater than that of the East Coast, reflecting a greater level of "chillness" among West Coasters.

Procedure

Part 1: Accessing the Twitter API and using Twitter4J library

To interact with Twitter and gain access to tweets sent by our target groups, we created several Twitter applications and generated Twitter API keys, access tokens and secret keys. One set of such keys is copied below as it appears on the Twitter Developers website.

Application Settings

Keep the "Consumer Secret" a secret. This key should never be human-readable in your application.

Consumer Key (API Key)	WzNkmENyonU6i94wlZzZUPPYg
Consumer Secret (API Secret)	LKFH0jf2rPtsFL2reCFo241Gs3QxsR3TlfWXr6LqLLD0CkBYVA
Access Level	Read, write, and direct messages (modify app permissions)
Owner	alikozlu
Owner ID	341538064

After receiving our credentials to access Twitter API, our group started looking through open source Java libraries to integrate our Java application with the Twitter service. Twitter4J is an unofficial Java library for the Twitter API. It is 100% compatible with Twitter API 1.1 and works on any Java platform. After adding twitter4j-core-4.0.3.jar to our application classpath and learning how to use the library by looking through its Javadocs and source code, we were ready to setup the connection.

Part 2: Setting up the connection

In this part we focused on creating a successful connection to Twitter API 1.1. We created a Setup class, which was responsible for creating necessary access to Twitter API for our purposes. In pseudo code, it works in the following way:

- Receive a ConfigurationBuilder object as input.
- Set up the necessary authorizations for our ConfigurationBuilder.
- Return a Twitter object for the other classes to use.

```
// Receive the ConfigurationBuilder object
cb = new ConfigurationBuilder();

// Set the necessary authorizations. Make sure we have all the necessary
// connections (such as being able to receive JSON Data.)
cb.setDebugEnabled(true);
cb.setOAuthConsumerKey("WzNkmENyonU6i94wlZzZUPPYg");
cb.setOAuthConsumerSecret("LKFH0jf2rPtsFL2reCFo241Gs3QxsR3TlfWXr6LqLLD0CkBYVA");
cb.setOAuthAccessToken("341538064-puQUKQe2yQOi349GqFzFpgjrHQ7MXOx3iVBUckqJ");
cb.setOAuthAccessTokenSecret("mLbzrul1G8RsDYVbbaHEoXTMJXNhBRFVr21u4tVWzzzMa");
cb.setOAuthAccessTokenSecret(
        "mLbzrul1G8RsDYVbbaHEoXTMJXNhBRFVr21u4tVWzzzMa")
        .setHttpConnectionTimeout(100000);
cb.setJSONStoreEnabled(true);

// create and return a twitter object
Twitter twitter = new TwitterFactory(cb.build()).getInstance();
return twitter;
```

Part 3: Reading tweets from a specific geo-location

To test our hypothesis, we needed to access large amounts of tweets sent by users in a very specific geographic location. Our solution was to use a data class in Twitter4J library called GeoLocation. In each run of our program, we specified a certain geographic location and created a specific query with our geographic location. Then we used the Query class (within the Twitter4J library) to search for the tweets. The TweetReader class received the Twitter object from Setup class, so we just had to call the search method of our Twitter object.

-Create the Query:

```
// give the input
double lat = +47.60890;
double lon = -122.33700;

// determine the radius
double res = 5;

// mile or kilometer
String resUnit = "mi";

// determine the number of wanted tweets
int wantedTweets = 10000;
int remainingTweets = wantedTweets;

Query query = new Query().geoCode(new GeoLocation(lat, lon), res,
        resUnit);
```

-Search from Twitter:

```
QueryResult result = twitter.search(query);
```

Part 4: Reading the tweets in JSON format and statuses text format

As we read the tweets, we saved them in a hash table that mapped each tweet's unique ID to its text. This hashtable guaranteed that we did not receive the text of the same tweet twice. We then created a word map that mapped each word to its frequency using a hash map. We also received JSON data from each tweet that we read for cross-checking purposes.

Part 5: Storing the Tweets in JSON format and text format

We created a Logger class and used the Observer - Observable Pattern to write the text of each tweet to a text file we called "testfile.txt.". Our program design made sure that every time we read a status, the text of that status was added to a text file. We used this file to test the map we created for our hypothesis. JUnit test case was written to make sure we mapped the right word, frequency map. The results are represented in the last part.

Part 6: Creating the main method

We created one set of words we deemed “chill” words, which tend to have mild, positive connotations, and one set of words we deemed “neurotic” words, which tend to have more passionate, negative connotations. This was done in a Word class.

“Chill” words:

- cool
- chill
- sweet
- nice
- fun
- lovely
- kind
- mellow
- woah
- ok
- okay
- awesome
- yay
- yes
- great

“Neurotic” words:

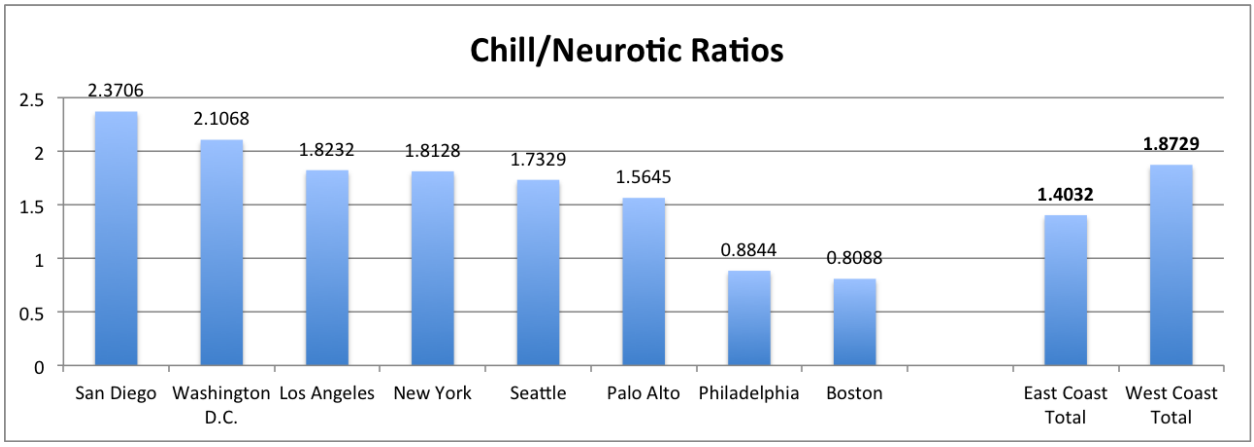
- omg
- wtf
- ugh
- hate
- wow
- seriously
- no
- awful
- terrible
- unbelievable
- worst
- disgusting
- crazy
- insane

We selected four cities on the East Coast (Boston, New York, Philadelphia, and Washington D.C.) and four cities on the West Coast (Seattle, Palo Alto, Los Angeles, and San Diego) and retrieved the most recent 2,000 tweets within a 5-mile radius from each location. The main method produces a count of each word in each of the two word sets and calculates the "chill/neurotic ratio." This ratio is equal to the number of occurrences of "chill" words found in the past 2000 tweets sent within a 5-mile radius from a given location (specified by latitude and longitude points in TweetReader.java), divided by the number of occurrences of "neurotic" words found within those tweets.

We found the longitude and latitude of the major cities using this link:
http://www.golombek.com/locations.html

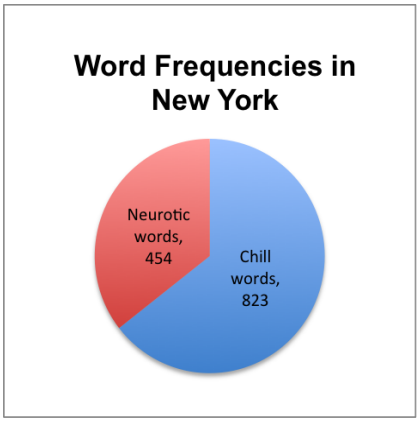
Please see our attached code for a raw version of our procedure.

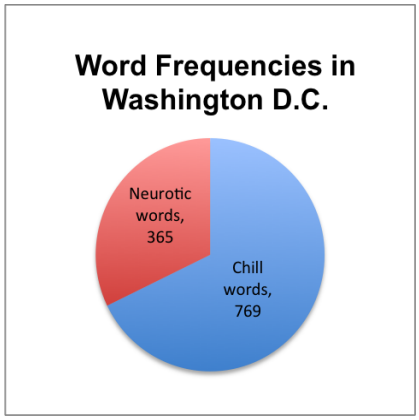
Results

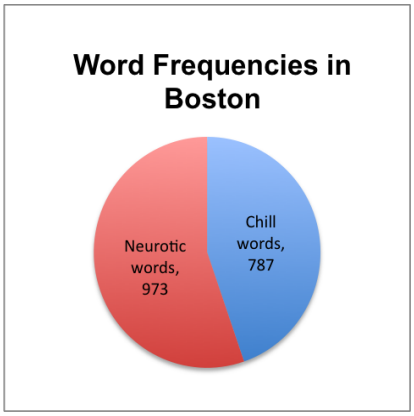


The above table depicts the chill/neurotic ratios that we found for each city and the average chill/neurotic ratios for each coast (found by averaging the ratios of the cities on that coast).

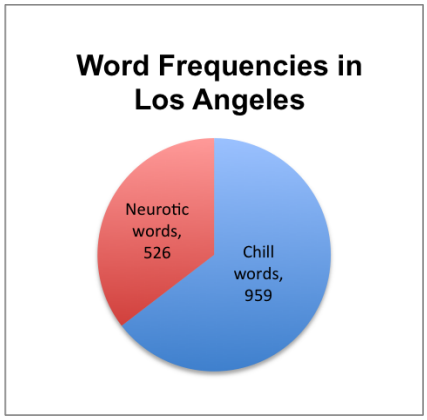
Word frequencies for four East Coast cities:

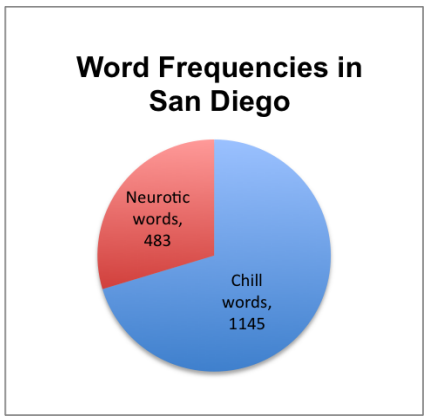


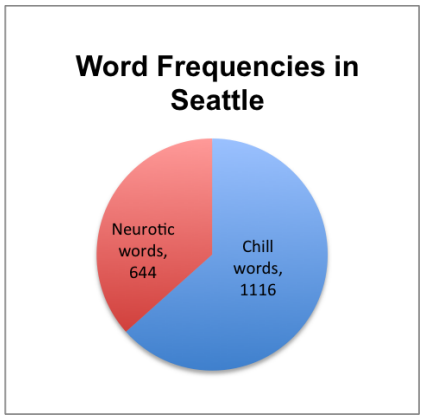


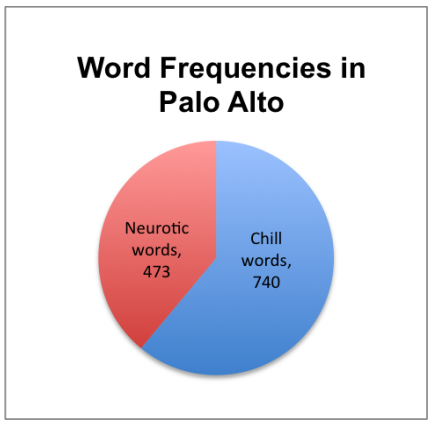


Word frequencies for four West Coast cities:

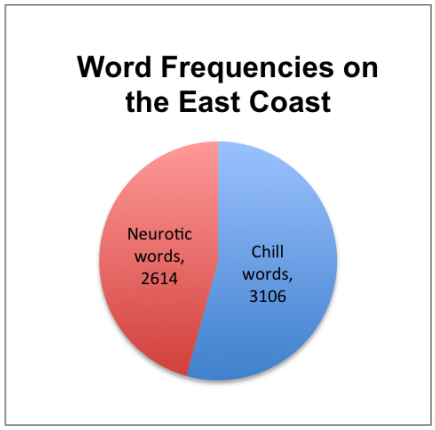


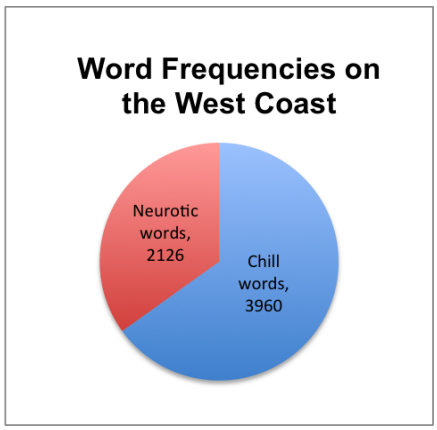






Total word frequencies per coast:





Please see “results.txt” for the raw data, containing individual word counts. “Results.txt” is a text file we created manually by compiling the console outputs of different executions of our main method.

Analysis and Conclusion

We found that the average East Coast chill/neurotic ratio (1.4032) is lower than the average West Coast chill/neurotic ratio (1.8729). Therefore, even though both coasts use "chill" words more often than they use "neurotic" ones (most likely because the "chill" words we chose were more common), the frequency with which West Coasters use "chill" words versus "neurotic" ones is larger than that of the East Coast. This data therefore validates our hypothesis.

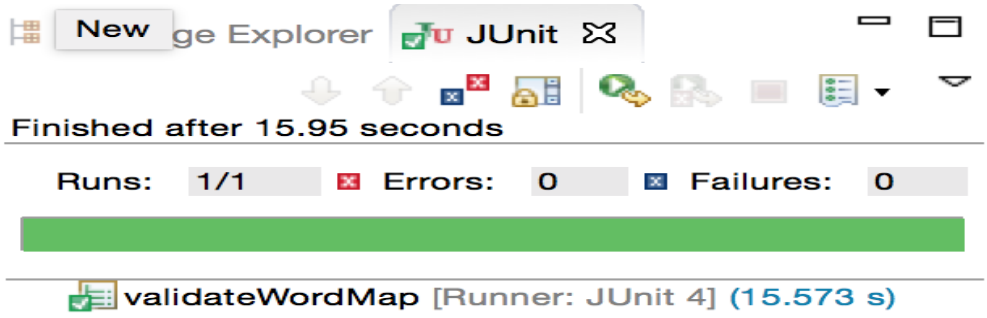
However, the second part of our hypothesis--that tweet content (particularly with respect to the words we selected) reflects an underlying "chillness" discrepancy--cannot be validated or invalidated by this experiment. As is the case with any data analysis, we are only able to observe correlation--not causation. That said, the fact that tweets sent on the West Coast contain more "chill" words does not mean that West Coasters are, in fact, more "chill." We acknowledge that one's words, let alone one's tweets, do not necessarily reflect one's state of mind. Nonetheless, we now know that West Coasters are *at least* consistent with their speech and their stated attitude.

Another flaw with our experiment is the data size. Because of restrictions on how many calls we could make to the Twitter API, we had to limit our data size to 2,000 tweets per location and only eight locations (even with the four separate developer accounts that we created). Also, Twitter does not grant developers access to historical data but rather only the most recent tweets in a given location. One way in which this restriction could have affected our results is that those on the East Coast (who are closer to the riots) may be responding more frequently to the current events in Baltimore. This may imply that East Coasters are using more negative words (which appear on the "neurotic" list) and less calm or positive words (which appear on the "chill" list) than usual. Perhaps if we performed this experiment at a different time, such as immediately following an earthquake in California, we would have obtained different results. One way to mitigate the effects of these data set restrictions is to create more developer accounts (which we could not do given our resources since each developer account must be linked to a verified phone number) and then rerun the experiment using more locations, larger radii (although not too large because then the data would extend too far inland), and more tweets per location. The results could also be further substantiated by performing the experiment multiple times across a long time period. The restrictions placed on the developer accounts are permanent, i.e., there is a permanent limit to the amount of information that can be accessed--not a limit on the amount of information that can be accessed given a different time period. Thus repeatedly performing the experiment across time would also require more developer accounts.

Part 7: Verifying our results:

To verify that our Word Map is correct, we created another Word Map in the ReadFile class, which parses through each line of the text file that contains the 2000 tweets in raw text format and uses the same algorithm to create a Word Map. We created a J-Unit test case that compares

the value of the key “great” in each map, and asserts they are equal. This was a way to ensure that we are getting the correct results.



The Data Files:

After running the program, the tweet texts can be found in the local file testfile.txt. Also the JSon data file that is created can be found under local file twitterjsondata.json. After formatted properly, a user in our JSon file has the following format in the document.json file.