Resolving Ambiguity in Product/Brand Names

Evie Gillie, Saahil Shenoy, Corey Stein
CS 341 Final Report

Problem

With over 140 million active users generating over 340 millions tweets and 1.6 billion search queries per day, the Twitter network contains a significant amount of information about what people are thinking, feeling, and doing. Such information is highly valuable and can be useful towards many different applications. One such application is extracting people's opinions about a particular product or brand. Companies and other organizations have a strong incentive to determine how people are responding to their products and actions, and to determine how they can improve their image. Traditional methods in accomplishing this have included surveys and focus groups, but such tactics are limiting due to the difficulty in getting large numbers of participants. But by using twitter data, one has access to the unfiltered, genuine opinions of millions of people across many different demographics.

However, a key element in trying to determine what people are saying about a particular topic is determining a set of tweets that are indeed about that subject. While there are times when querying for a product name will suffice, there are many instances where such a tactic won't work. The following table lists a few such examples.

Keyword	Subject 1	Subject 2	Subject 3
Reese	Candy	Witherspoon	
Colgate	College	Toothpaste	
Giants	New York	San Francisco	
Simpson	Jessica	Cody	TV Show
USC	University of Southern California	University of South Carolina	
Tigers	Detroit	Louisiana State University	
Stern	Howard	David Stern, Commissioner of NBA	
Cardinals	St. Louis	Arizona	
Trojan	University of Southern California	Computer Virus	Condoms
Eclipse	Programming Environment	Twilight Series	Solar/Lunar Eclipse
Cowboys	Dallas	Oklahoma State	
Bruins	UCLA	Boston	

That is, if we want to find all tweets about Colgate the college, and collect a set of all tweets containing the term "colgate", only some fraction of that set will actually be about the college, and the rest will be about the toothpaste, and perhaps some other less common meanings. When one encounters this issue of ambiguity when searching on a different database, such as Google, the search can easily be narrowed by adding some manually chosen keywords ("college") to the query. However, this same approach does not work when examining Twitter data. Because tweets are limited to 140 characters, they are very short and concise and when accounting for stop words, spaces, and punctuation, each tweet contains very few words of substance from which to determine the subject matter. Therefore, if one wants to find tweets about Reese's the candy, trying to find all tweets that contain the words "reese", "peanut butter", and "cups", there will be very few tweets to look at.

Simply put, our project was aimed at resolving this ambiguity through automated processes.

Data

Twitter generously provided us with a very large dataset that contained over 3 billion tweets from late September – late October 2011. In addition to the text of the tweet, other fields were also provided including the date, time, number of followers, and geo-location. In total, the English language portion of the dataset was approximately 600GB.

Generating Related Keywords

Our first step in trying to determine a set of tweets about a query term was to find a set of related keywords, which we will call "indicator keywords", as they indicate that the instance of the query term does indeed have the meaning we want.

We initially explored using tools from the internet such as Google's "Related searches" tool, but found that such methods were ineffective and did not extend well to tweets. Instead, we decided to use the texts of the tweets in order to generate related search keywords. The measure we created for ranking possible keywords was developed from our knowledge of association rules. Specifically, we considered each tweet to be a "basket" of individual words. Then, given a target word *t*, we ranked any possible related keyword *k* by the following similarity measure:

$$\mathrm{similarity}(t,k) = \frac{\mathrm{support}(t \cup k)}{\mathrm{support}(k)}$$

where support(*x*) means the number of tweets that contain the word *x*. This measure of similarity makes sense because if a possible keyword *k* occurs often with the target word *t* then they are more likely to be related, and so the score should increase. Similarly, if *k* occurs more often throughout the entire set of tweets, then it is more likely to be a popular word and not necessarily related to the target word *t*.

Overall, this measure worked very well and for many products and brand names we were able to generate a good set of related keywords.

Determining Relevant Keywords: Two Approaches

After devising a system to come up with relevant keywords, we had to tackle the issue of determining keywords for terms that were ambiguous. For example, the top fifteen keywords for Reese were:

witherspoon, toth, pieces, cups, twix, peanut, puffs, kitkats, butterfingers, butter, snickers, hersheys, kats, twizzlers, skittles

While the first two keywords relate to Reese Witherspoon, the last thirteen are targeted at the candy (toth refers to Jason Toth, the husband of Reese Witherspoon). We came up with two ways to resolve this problem.

Method #1: Using Categories

Our first method relied on the fact that items sometimes belong to specific, well-defined categories, and Walmart had given us a well-defined hierarchy of items that they sold. For example, Reese's Peanut Butter Cups clearly belongs to the category of "candy", and we know what other items belong to the category of "candy" in the Walmart data. Perhaps we could extract features of the "candy" category, and match our "reese" tweets against those features.

The first step was to generate a "master list" of features for a given category. Given a list of items: {item_1, item_2, …, item_n} in a category, we found all tweets containing any of those item names. We will call this set of tweets *T*. Then, we removed stop words and ranked terms present in *T* by tf-idf. This generated a reasonable list, but we found that some high-ranking words referred to a single item's other meaning – for example, reese's non-candy meaning introduced the term "witherspoon", which had a high tf-idf and ranked high. In order to downplay individual items' non-candy meanings, we assigned terms a score of tf-idf *(the number of distinct item names they appeared with in *T*).

The second step was to rank the terms occurring in tweets containing the query word ("reese") by tf-idf. We separate the top n terms in this list (an n of 15 seemed to generally word), and filter out any words that did not occur in the top n*2 terms from the category master list. This successfully filtered out non-candy terms, but also filtered out some item-specific terms that only appear in one candy. For example, it filtered out "cups" from the "reese" terms, since "cup" only ever occurs.

Method #2: Clustering Key Words

Although the first method worked well for candies, the assumption that all items will have as well defined a category was a strong one and was not easily extendable. However, we also came up with a second method that works in a more general setting. Our second method centers around the idea that the keywords that are related to the intended meaning of the query word will probably be closely related to each other. For example, taking the top words that co-occur with "reese", one would expect that the words "snickers" and "peanut" would be more closely related to each other than either is to "witherspoon".

We used jacquard similarity as our notion of similarity. That is, given keywords k_1 *and* k_2, and support(x) meaning the count of tweets containing the term x:

$$\text{Jacquard Similarity}(k_1, k_2) = \frac{\text{support}(k_1 \cap k_2)}{\text{support}(k_1 \cup k_2)}$$

Using this notion of similarity, we were able to construct an undirected, complete graph with the nodes representing the keywords and the edge weight equaling the Jacquard similarity between the two keywords.

Our plan was to then cluster the nodes of this graph, with the end result hopefully being that each cluster would represent a possible meaning of the targeted word. However, all of our experiments yielded distinct components. With the number of tweets we considered (approximately 1 billion English tweets), words like “Snickers” and “Witherspoon” don’t occur together. Since the number of words of substance in a tweet is so few, unrelated words tend to occur together either never or very rarely. However, if a larger number of tweets were to be considered, it would seem that a clustering method would probably apply.

We tested this method on several different ambiguities and received very positive results. Outside of Reese, other examples include “Tigers” and “Stern.” During the timeframe of the data, the Detroit Tigers were competing in the Major League Baseball Playoffs, while the Louisiana State University Tigers were the top ranked collegiate football team. When we applied the clustering method, to our surprise, we actually had 3 different components: one for baseball and football, as well as one based on tigers the animal (which included words like “lion” and “cheetah”). For “Stern,” we found two distinct clusters, one for Howard Stern and one for David Stern, the commissioner of the National Basketball Association. Although Howard Stern would generally be considered more famous, during the timeframe of our data the NBA was in a lockout, and so David Stern was receiving a lot of attention on Twitter. The fact that this method worked towards a product name, a brand name, and a person’s name shows its flexibility and wide-ranging applicability.

Precision and Recall Using Method #2

Having decided to pursue method #2 over method #1, we next investigated the trade-offs of certain parameters, namely *n*, the number of top co-occurring words we choose as indicators that a tweet is about the intended target.

Given a target query “tigers”, we found the top 45 co-occurring terms for the baseball and football clusters. We then constructed sets of tweets that each contain “tigers” and any of the top 5 keywords using method #2 from the baseball or football cluster. We then sampled 100 tweets out of the resulting set and manually recorded how many were about the appropriate team. We then repeated this process for the top 10, 15, ..., 45 keywords to build a precision/recall graph

12650 tweets included “ tigers “
Breakdown (based on random sampling and a lot of tweet reading):

- 33% Detroit baseball
- 27% LSU football
- 40% neither topic

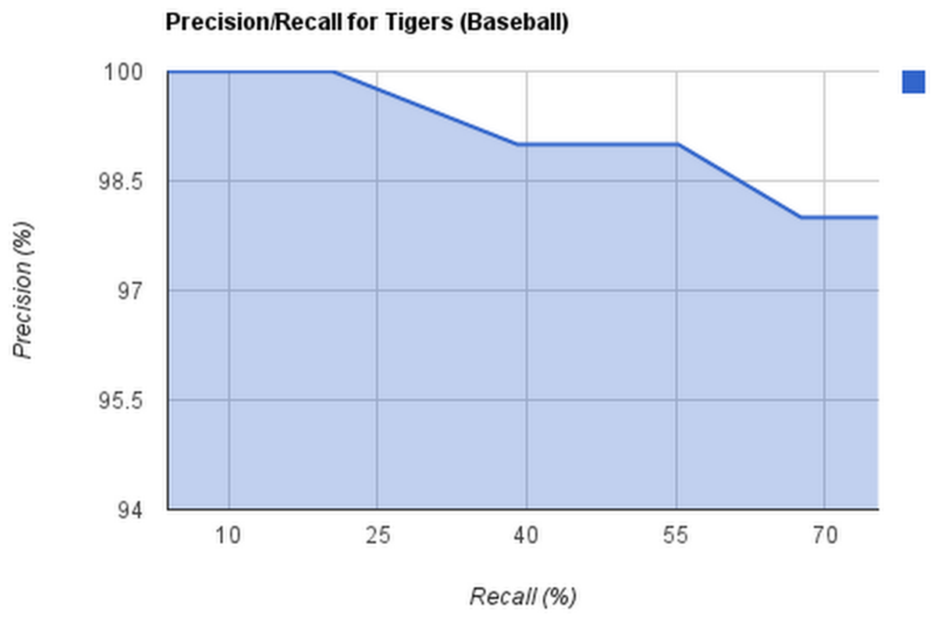
Detroit Baseball

Number of Keywords	Number of Tweets	Precision (%)
5	160	100
10	292	100
15	852	100
20	1630	99

25	2036	99
30	2307	99
35	2823	98
40	2911	98
45	3147	98

LSU Football:

Number of Keywords	Number of Tweets	Precision (%)
5	2075	98
10	2196	97
15	2275	97
20	2547	95
25	2723	94



Precision is out of an estimated 4200 tweets

Extensions

Now that we can filter a set of tweets down to only those tweets we are confident are about the target topic, we can work on expanding our set of tweets. In the methods above, we always started with the set of tweets containing some keyword, such as “cardinals”. What if we could find all tweets about cardinals that don't contain the keyword itself? We did some preliminary work with synonym finding, wherein we devised the following algorithm:

```
FindSynonyms(W):
    T1 <- all tweets containing W
    S <- top n words in T1 after tf-idf
    T2 <- all tweets containing any words from S
    S2 <- top n+k tf-idf words
    Return the word that co-occurs with the most words from S
```

Out of 7 tests, this yielded one false positive (a query over "cardinal" returned "series" instead of "cards").

Another significant extension would be to implement multi word phrases. So far the analysis done above has been for single word queries on the Twitter dataset. A possible way to extend this is to apply the same techniques as before but run queries on bi- or tri-grams.

To the end of predicting Walmart sales, we could also add sentiment analysis to score the body of tweets about a topic. We tried a simple scoring system using the word tags at: http://www.cs.pitt.edu/mpqa/subj_lexicon.html but would ideally try more refined sentiment analysis.

Concluding Thoughts

Once again it has been shown empirically that by having a large amount of data (large in the sense of the size of the Twitter data) much simpler algorithms tend to work more efficiently than more complicated and elaborate methods. Also the graph of precision against number of keywords behaved as predicted, i.e. in a decreasing fashion.