$ale Cloud

Final Report

Team

Vimal Kini - UX, Coding, Research
Divya Karthikeyan - Project Reports, Coding, Quality Analysis
Gaurav Nitin Shetti - Coding, Research, Product Management
Arian Shams - Project Manager, UI, Coding

Mentor

Miguel Rios

Project Goal

Visualizing sales/deals over Bay area and SF on the basis of Tweets. The project would encompass

1. Using Twitter's API to detect tweets about sales/deals from SF/East Bay in near real-time
2. Visualizing tweets related to sales on map as a "sales cloud" to allow users to instantly identify hot sales/deals that other shoppers are tweeting about.
3. Updating the map in real-time (about every ½ hour)
4. Allowing users to search by topics within the tweets and adjust the map accordingly. For instance, a user can search for tweets related to "Shoes," which would then modify the map to show tweets relating to sales with the term "Shoes" in the text.
5. Allowing users to focus on certain regions of the map and view the sales tweets from those regions.

Project Timeline, Milestones and Strategy

Date	Task	Deliverable
Oct 26 **(Completed)**	Finalize Project Scope.	Setup project website
Nov 2 **(Completed)**	Complete research relating to filtering tweets against a certain topic (sales for us). Also, finalize front end user interface prototype.	Front end prototypes and summarization of research
Nov 9 **(Completed)**	Complete analysis of tweets relating to sales and develop algorithm for filtering sales related tweets.	Preliminary algorithm for filtering tweets
Nov 13 **(Completed)**	Have a functioning front end interface that is tested and works with the defined data structure that will feed into the interface. Also, create a preliminary design of how the backend servers will automatically capture tweets, filter them and then transform them to a structured data type that will feed into the front end.	Working site that maps data to an interactive map. Preliminary design of backend server setup
Nov 23 **(Completed)**	Finalize testing of algorithm that filters tweets relating to sales.	Finalized sales filtering algorithm.
Nov 30 **(Completed)**	Finalize font end user interface. Finalized implementation of backend servers that will capture, filter, weigh and transform sales related tweets to the proper data type to feed into the front end. Finalize entity extraction algorithm.	Functioning website that meets criteria listed in project goals
Dec 10 **(Completed)**	Quality Testing and modification (if needed). Complete write-up.	Project Complete

Grading Criteria

A = We have met all or all but one of the criteria listed in the Project Goals. The filtering algorithm that filters for sales related tweets has a reasonably low error rate and the front end interface is easy to use and provides value to the user. Most important of all, the final deliverable helps people find sales based on tweets in real-time.

B = We have not met two or three of the criteria listed in the Project Goals. The filtering algorithm has a high error rate (but is reasonable). The final deliverable helps people find tweets related to sales but many (if not most) of the tweets are not sales related.

C = We have not met four or more of the criteria listed in the Project Goals. The filtering algorithm has a high error rate that is unreasonable and does not help provide the proper tweets for the user to identify sales. The front end interface is not easy to use and does not provide value to the user. The final deliverable does not help the user identify sales due to unreasonably high error rates in filtering tweets and a poorly (or non-functioning) front end user interface.

Design & Development Process

Original Idea

Our original idea of this project was to aggregate real time geo-tagged tweets from consumers and visualize them over a map. This idea was based on 2 assumptions;

1. People tweet when they find a great deal at stores

We used Twitters streaming API to collect tweets over a period of 4 days including the weekend. We observed that our assumption about users tweeting when they find a great deal at stores does not hold true. We found an insignificant number of tweets from users tweeting about deals they found at stores. Another challenge was that users did not necessarily tweet from the location where they found the deal. Users could be tweeting from their homes about a great deal they found during their shopping trip.

2. We can get a substantial number of geo tagged tweets.

We filtered Twitters streaming API to only output geo-tagged tweets within the San Francisco Bay Area. Again we found very few tweets that were related to sales and deals at stores. It was not clear if this pattern was specific to San Francisco or generic. By tracking geo-tagged tweets we were also missing out on tweets from users who did not have their geolocation enabled.

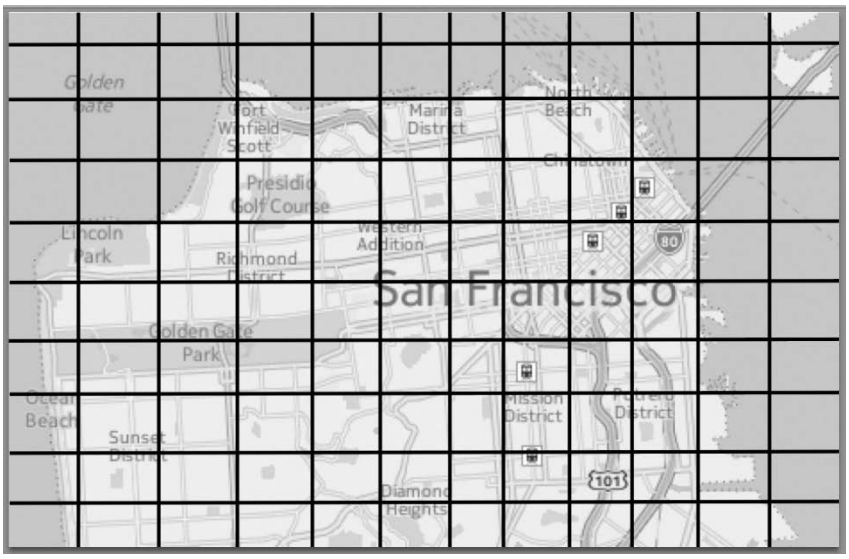
We tried a couple of solutions to overcome these challenges.

Solution A: Raffle for useful user tweets

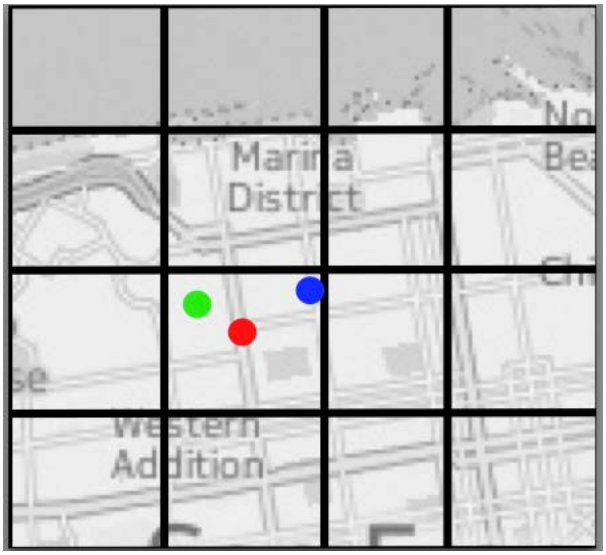
One of the ideas was to create an eco-system that would incentivize users to tweet more. The plan was to program a bot that would look for tweets relating to deals and sales. If there was no store locations associated with the tweet, the bot would reply to users and ask for the store location where they found the deals. The bot would also inform users that by replying they could enter a raffle and win a prize. However, we were not sure if users would trust and react positively to a tweet bot, especially if it were asking for a specific map location of a deal. There were additional questions around what would be a good incentive for users to tweet about deals, such as whether to enter a long term raffle or to receive a quick reward like a coupon. Another idea was to create a leaderboard where shoppers who found the most number of good deals could be rewarded with reputation points. The team decided that figuring the best incentive model would need more user research, which was not possible with the limited amount of time for this project. Hence, we switched to an alternative programmatic solution.

Solution B: Geo-Clustering

If we were going to plot all the tweets on the map, there could have been a case of discrete data points being plotted everywhere on the map. The users would have to click individually on each of these data point to read the tweet. It would have been a potential pain point for the users. Hence we needed an aggregation service to cluster these individual tweets within a particular vicinity. Our mapping.py module achieves this functionality. This script initially accepts 2 extreme end points on the map - namely the diagonals (top left and bottom right). It then divides the map into squares of equal sizes (we tested with 1000 rows * height). This is done during initialization.



Once the image is divided into smaller squares (constructor is set), the algorithm accepts x,y coordinates of the points to be plotted. It then checks which square the x,y point plots to. Once identified, it returns the midpoint of that square. Thus, in this manner, if we have 2 or more tweets falling within a smaller square, they both would point to the same coordinates



For instance if the green and blue dots in the above figure represent individual tweets, after passing it to mapping.py, it would return us the red-dot as the new location of the tweet. Thus, the users would just have to click on the red dot to get all the tweets in the nearby region. This potentially solves the pain point of clicking on *n* random sale/deal related tweets. The logic of the code can be found at: https://github.com/gauravshetti/salecloud/blob/master/mapping.py

In spite of clustering the tweets together, we did not receive a substantial number of geo-tagged tweets. Our team mentor also expressed concerns about relying on geo-tagged tweets via the twitter API for real

time tweets since he expected the number to very small. Hence the team decided to pivot to a different model to visualize tweets.

New Idea

We decided to split the tweet analysis into two parts:

1. Entities Section
Search API fetching sales/deals related to tweets from the SF bay area (Either geo tagged tweets or tweets from users whose profile location is set to be in SF)

2. Store Map Section
User Timeline API to retrieve tweets from the timeline of specific stores that often tweet about in store deals and sales that are time sensitive

For the Entities Section, we decided that a feasible solution is to display the data using a bubble interface. The size of the bubble varies based on the popularity of the tweet. The number of times the tweet is re-tweeted and the number of followers of the tweet owner are the primary criteria in calculating the weight for popularity.

Tweet Analysis & Filtering

The filter criteria used for the Entities Section and the Store Map Section were slightly different to suit their needs.

Entities Section
Hash tags such as #deal, #sale, #discount, #price and specific keywords like deal, sale, discount etc. were used as part of the search query during the API call. The Search API also filtered tweets for specific geolocation tagged or user profile location data. Further analysis on the tweets were performed inside Python code where its principal purpose was removal of false positives and ensuring at least 80% precision. Some example criteria used are:

- To remove forms of the term 'sale' like sales force, salesman, sales representative, sales team etc. by verifying if the word sales occurs in a phrase along with any of the above terms.
- To remove forms of the term 'deal' like deal with, deal between and various other false matches containing deal by
  - Testing for the occurrence of $ in the tweet
  - Examining word phrases looking for with, between etc. occurring along with deal
- To consider only specific forms of offer like 'special offer.'

The filtered tweets were given some weightage before being passed on to the UI for visualization. The weightages are based on 2 aspects:

1. Re-Tweet count
If a tweet related to sale or deals is re-tweeted, it must be popular. Twitters Search API provides a re-tweet count attribute for every tweet returned. We added re-tweet count linearly to the weight.

2. Followers count
The intuition behind using follower count as the weightage was that the higher the follower count, the more credible the tweet would be. But we also observed that a high number of follower counts in the range of 100,000's could significantly bump the weightage and there was

also a higher chance of that tweet being a false positive (extremely influential people may not care so much about sales and deals and may be talking something else). So we scaled down the follower count by a factor of 1000 before adding it to weight, which seemed to show a good balance on the visualization.

Store Map Section

We manually curated a database of stores that had active twitter accounts and tweeted regularly about sales and promotions along with their locations. We pulled the user timelines from all the store twitter accounts and filtered them for sale and deals related tweets.

The filter was much simpler here as the tweets posted by stores are mostly pertaining to store products, sales, deals, offers, or new product releases. Hence the filter contained:

- A basic search for deals or sales. The false positive removal was not required here as there were no conversation tweets on stores, rather just announcement or advertising tweets.
- Included a clause to filter out "free shipping" as we were particularly interested in In-Store deals only. Free shipping would refer to online deals posted by stores

Entity Extraction

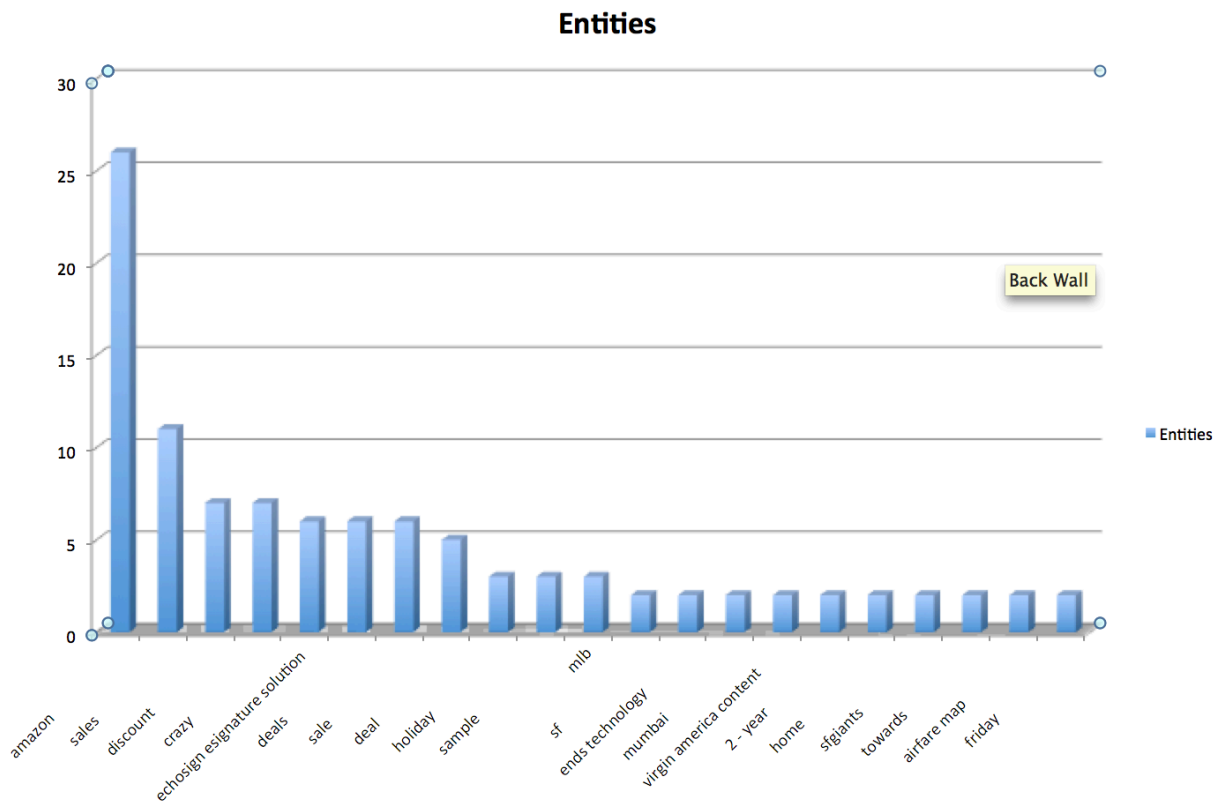
For the *Entities Section*, we decided to develop our own algorithm to identify entities rather than bank upon LDA (Latent Dirichlet Allocation) because of the following reasons:

- The accuracy of LDA is good only for a certain *n* categories
- Tweet text is very limited and hence the accuracy of machine learning algorithms tends to vary a lot
- Our scope was narrowed down to deals and sales and hence the structure of most of the tweets would be similar.

We took a lot of cues from the technical paper on 'Extracting events and description from Twitter' [12]. This paper banks on the parts of speech tagging and uses grammatical concepts on the underlying structure of English sentences. Similar to the paper, we applied the nltk's parts of speech tagger and researched on how to extract nouns relevant to the scope of this project.

Explanation & Global Pool

We used a global pool of previously identified topics to facilitate clustering of documents into one topic. We boosted the weightage of words that matched this global pool of topics (*weight of 10*). This pool of topics grows gradually with time. The administrator can also manually curate the topics if required. For **201 tweets** which we collected over a period of **2 hours on 7th December, 2012**; the top results were as follows:



- 21 topics converged out of 127 topics. Statistically 17% of the total topics converged. Considering the outliers, we got 1 out of every 5 tweets to converge to a topic, which is good since no machine learning algorithm was used.
- 103 out of 201 records converged into a similar topic i.e 51% of the tweets converged. The above statistics are good since the algorithm warms up with the initial trending words and then converges the tweets.
- We also maintained a list of stop words or words which were not meant for topics. These are manually curated over time.

Placeholders

- Since urls, user mentions are valid nouns but don't count as valid topics, we replaced them with placeholders.
- Hashtags were identified initially and were given a weight of 6. This bias was 60% of the weight assigned to global tags since we wanted to bias the hashtags more than just 50% of highest weightage. After identifying hashtags and adding them to entity dict, they were eventually replaced.
- Discounts (keywords with %) are identified and recorded in entities dictionary with weightage of 1. The respective placeholders eventually replace them and the urls.
- Person mentions were also replaced. However there were cases where places were tagged with @ signs. These were obvious stores with registered twitter profile names. So we had 2 regexes to identify these unique scenarios.

```
loc_topic_at = re.compile(' at @([\w]+)')
```

The above regex searches for keyword 'at @' immediately followed by a location reference. Though we can extend it to keywords like 'in','next to' and so on, the number of false positives were high for those keywords. We factored this as future studies and would like to explore it after the end of the class. Keywords found by this regex gets the highest priority and is termed as the entity for the tweet irrespective of the weightage of words in the entity dictionary.

```
loc_topic_at2 = re.compile(' (sales|sale|deal|deals) @([\w]+)')
```

This gets the second highest preference after the first regex. Entities found by this method also get accumulated as the highest priority topics irrespective of the words/topics in entity dictionary. This regex finds keywords like sale, sales, deal, deals with @ keyword. They are probably using @ for keyword 'at' since tweet text is limited.

Parts of speech tagging

- After neutralizing the tweet text with placeholders, the nltk library of python is used. The words are tokenized on the basis of spaces and punctuations
- These tokens are then passed through the parts of speech tagger of nltk library. These parts of speech are themselves trained over thousands of words from the brown corpus dataset. Hence they have greater than 80% accuracy.
- We used unigrams since it would be tougher to associate parts of speech (POS) to bigrams.
- The initial POS tagging contains all complex parts of speech as well. To reduce the dimensionality and to simplify these parts of speech, we simplified the tags.
  http://nltk.googlecode.com/svn/trunk/doc/book/ch05.html look at table 5.1
- We then analyze the tokens solely on the base of these tags. We do a forward lookup and a back lookup based on the keywords found.

Back lookup

- If the tweet contains any of these keywords (sale/sales/deal/deals), which it should atleast once, the code scans all the words from this breakpoint backwards to find the first noun, which are not the placeholders replaced above.
- Once a noun is found (proper noun/common noun), that is potentially the topic match. The reason being, many of the tweets contain sale tweets which have a defining entity right before the keywords sale,sales,deal or deals are mentioned. This keyword describes the product.
  For eg *"@bfunderburg, Hitman: Absolution is now on sale for $52.97 at Amazon. Product page: url"*.
  Here absolution would be the keyword that is the first noun found before sale.

- Once the keyword is found, we look for any additional nouns or numbers or any grammatical marks like (., ', etc) prepended to the found noun. This process stops till the first word of the sentence is hit or any other grammar tags are found except for the ones explained above. This collection of words together forms one of the entities, which are weighted with a weightage of 4. The reason for 4 being, it is slightly less relevant than hashtags but doesn't deviate from the actual topic of the tweet, had the POS tagger been 100% accurate.

Forward lookup

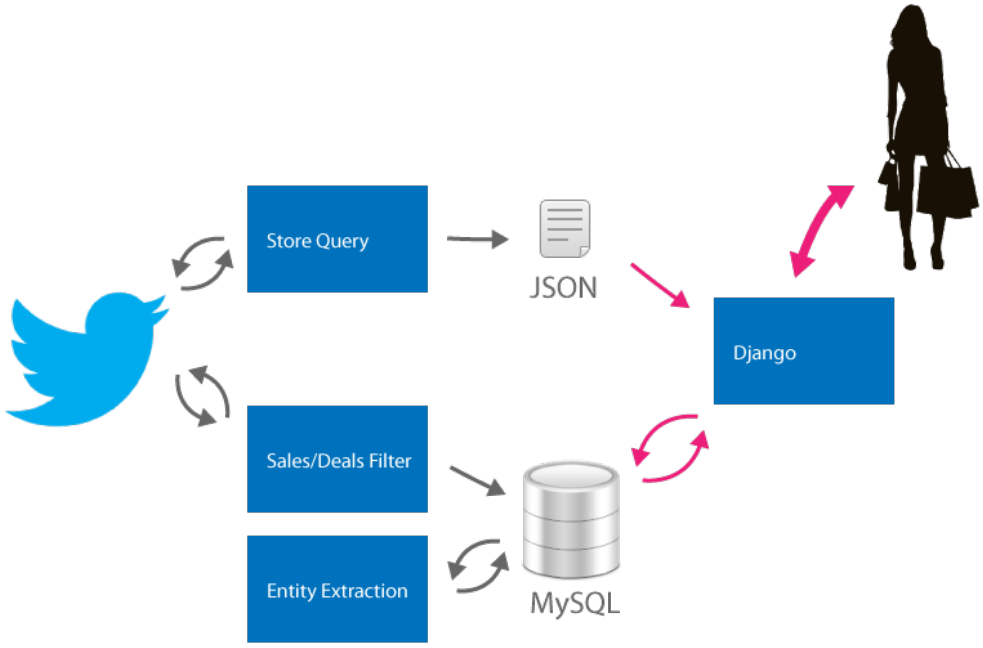
- Once keywords like deals, at, off, with, on are found, the breakpoint is set to the next word and we immediately look for nouns
- Once a noun is found, we look for any additional nouns, numbers or grammatical marks like hyphens, quotes etc.
- Its an iterative process and this keyword/collection of keywords is added to entity dictionary.

- If the noun is found after the breakpoint was met with keyword ‘at’, the entities found would have a weightage of 6 else it would have a weightage of 3.
- Reason for weightage 6 is because the location makes much sense for a deal tweet whereas for others, it’s not a strong indication of the topic of the tweet.
- If a full stop is encountered, we check the previous word to see if its length is 1. If yes, we append the words on either sides of the period. This is a way around not to miss abbreviations, like J.Crew

Final Words

- The words found through each of the methods above are put in an entity dictionary. If the same words are present in the global pool, it is boosted by a weight of 10. The entity dictionary is then sorted in the descending order of their weights. The first element of this dictionary is then fixed as the topic for the tweet.
- To restrict the topic size, we manually count the number of words and limit it to less than 3 words for each tweet.

Architecture



The figure above shows an overview of the SaleCloud architecture beginning (on the left) with retrieving sales and deals related tweets from Twitter and ending (on the right) with a UI that is delivered to the user via the Django web framework. The backend architecture is divided into two main parts: (1) the Store Map Section, and (2) the Entities Section.

Store Map Section

The Store Map Section is responsible for querying each registered stores Twitter timeline, filtering all tweets relating to sales and storing those tweets along with the stores geolocation in a JSON file which can be retrieved by

the UI and plotted on the map portion of the interface. Currently all stores are manually registered but we have added a feature where stores can sign up and be added in our registry by "following" our twitter handle (@SaleCloud). When a store follows our Twitter handle, we will need to verify that they are in fact a store and add their twitter handle along with their geolocation into our store registry. Twitter's search API is queried through Tweepy every five (5) minutes. The system adds store tweets directly to a JSON file rather than a database since a static file can be retrieved and served to the user much quicker than a dynamically generated JSON object.

Entities Section

The Entities Section is responsible for querying for the most recent tweets submitted by random users within the San Francisco area, filtering all tweets relating to sales, extracting entities from the tweet texts, weighing and ranking each entity based on the users search criteria and returning a maximum of fifty (50) entities along with each entities tweets to the UI which is then visualized through a d3 Pack Layout. The Entities Section is slightly more complex task than the Map Section in that it has to filter through Tweets generated by random users and analyze each tweet so see whether an entity can be extracted from the tweet text, which is also an indication of its relevance. The first step in our architecture is a Twitter search API call through Tweepy every five (5) minutes for tweets relating to certain keywords such as "sales" and "deals" *since* the most recent tweet that was collected by our system, which we can do by using the highest tweet ID stored in our database. In this way we ensure that we are searching for the most recent tweets beginning after the latest tweet we have in our database. Those tweets along with a variety of attributes are then dumped into a MySQL database where we have a separate process go through and attempt to extract entities from the tweet text. In addition, each time we capture a tweet that is re-tweeted but already stored in our database, we a record that we captured a re-tweet for the respective tweet along with the followers count of the re-tweeter. All tweets have a full text index on the tweet text to allow for fast querying. When a user submits a query (i.e. searches on our site), the database searches to find tweets that have been processed by the entity extractor and that have any of the words within the tweet text. It then returns a subset of the tweets along with their corresponding entity. Those results are then ranked and weighed using an algorithm that favors "popular" tweets by looking primarily at the re-tweet count and the followers count for each particular tweet. Those weights are then aggregated for each entity and then ranked and returned to the UI to be plotted via the d3 Pack Layout.

UI & Frontend

The visualization tools we used in the UI primarily consist of Leaflet for the map portion of the interface and the d3 Pack Layout for the entities portion of the interface. We also used jQuery and Underscore for some basic interactivity and data processing. The UI is set to poll our server every five (5) minutes to re-plot both the map points and the entities respective to what the current search term the user has entered. We used Django as our web framework to process all requests and return the appropriate JSON response to the UI.

Project Results

Overall we had some very good results with our project, both in terms of processing tweets and developing the visualization to show them. All of our code can be found on GitHub at: https://github.com/gauravshetti/salecloud. To access the site, you can visit www.tinyurl.com/salecloudd or http://people.ischool.berkeley.edu/~ashams/salecloud. In terms of the tweet filtering, as of December 9th, 2012 we measured a 79% success rate, which is in a range we were told was pretty good for Twitter text from our project mentor. Overall we achieved the following in relation to our stated goals:

(1) We successfully developed a backend infrastructure that queries the Twitter search API via Tweepy every five (5) minutes to gather the latest tweets since the most recent tweets that were previously captured (5 minutes was the minimum interval time before we hit the API rate limit). We chose to capture tweets by this method as opposed to streaming the tweets for two reasons: (1) if our running processes were interrupted then we would not lose any tweets during the interruption time, and (2) there are certain filters and status attribute results the Twitter search API offers that are not available from the Twitter streaming API. In addition to successfully developing a Twitter capturing backend, we modified the Tweepy code itself to use the latest API offered by Twitter in order to receive the maximum result sets. We are hoping to commit our changes to the Tweepy code for others to benefit from. For now they can be accessed at
https://github.com/gauravshetti/salecloud/blob/master/binder.py
https://github.com/gauravshetti/salecloud/blob/master/api.py

(2) We were able to visualize sales related tweet with two (2) visualization techniques. We first plotted points on an interactive map to show the locations of stores that are tweeting about sales and we developed an interactive bubble chart visualizing the most popular tweets relating to sales and deals for any given term that a user searches for. We feel that these two visualization techniques complement one another to create a unique “sale cloud” UI which allows users to explore Tweets to find hot sales and deals.

(3) We implemented an architecture that allows for quick data updates to the UI in real time with new data pushed every five (5) minutes or whenever the user searches or refreshes the page. We achieved this quick data push by implementing a proper full text index on our data along with limiting dynamically generated data to only calls that are not in a cache and that need to be dynamically generated.

(4) We implemented a search feature by which users can search for specific items (terms) in a tweet text and the points on the map would adjust to show only those dots that return positive results for that particular search term. In addition, a search would also fetch the top fifty (50) ranked and weighted entities containing tweets that in turn contain the search term. The size of the entities are then calculated in relation to the weightage of the total returned result set so that the user can judge which entity is the most popular for their particular search.

(5) Our map visualization allows users to navigation throughout the city of San Francisco to find stores in various regions that are tweeting about sales and deals. The UI map interactivity is based on the leaflet JavaScript library. Currently our project is limited in scope to San Francisco but it can easily accommodate other cities.

Future Work

If this project were to continue beyond this course, there are a few features we believe can help improve our SaleCloud system. Please note, we were not able to pursue these features due to limitations in time.

(1) To help improve our Tweet filtering algorithms, we can implement a UI feature where users would be able to mark which tweets are not related to sales and/or deals. If we have a large user base then we can potentially use that data as a training set for a machine learning algorithm that could better filter tweets.

(2) We can implement an up-vote/down-vote UI where users would be able to up-vote or down-vote each tweet as to how good a deal they are. This data would in turn be used to adjust the weightage of that particular tweet for subsequent searches by other users.

(3) We can implement a mobile app where users can sign in with their twitter account and then specifically tweet about sales and deals through our app, which would then be captured by our system in addition to being posted on their timeline. Our UI would then show a different set of points on a map, perhaps colored differently, that indicate sales and deals from regular twitter users. The incentive for a user to do this may be to potentially put their twitter account in a monthly lottery to win a small prize.

(4) We can improve the search functionality of the site by allowing the user to search by pre-defined categories which we could assign to each entity and/or tweet. We can also implement an auto-complete feature where the search bar is prepopulated by available search term choices that are extracted from our corpus of tweets. We can additionally add unique filters to tweets such as tweets originating between a certain time range or tweets based on a minimum or maximum re-tweet or followers count.

Work Percentage

Task	Arian	Gaurav	Vimal	Divya
UI	80		20	
Django Implementation	50		50	
Database Implementation & Search/Weightage	70	30		
Backend (Map Section)			80	20
Backend (Entity Section)		90		10
Tweet Filtering Analysis		20		80
Twitter API Research		60		40
Store Registry Curation			50	50
Integration	25	25	25	25
Literature Review	25	25	25	25
Final Report	25	25	25	25
Presentation	25	25	25	25

Literature Review

The research for our project can be broken down into three main categories. The first category has to do with the search terms used by users where tweeting about e-Commerce related topics such as “sales” and “deals.” The idea is to identify within some range of accuracy the various terms that users may use to describe tweets correlating with sales related events and then to filter or search for tweets that meet such criteria. The second category has to do with techniques for analyzing and mapping geo-tagged tweets. Specifically, we wanted to know how best to cluster geo-tagged tweets and to identify certain patterns within a geographic area in context to businesses located in that area. The third category of our research has to do with analyzing tweets, specifically in the context of lexical analysis, search and filter techniques, tweet categorization, and sentiment analysis.

Throughout our design and development process, we focused less on the geo-location related research due to our pivot in how we were to implement the mapping features of our project. Most of our project drew from research relating to eCommerce search terms and tweet analysis in that they helped guide our efforts to effectively filter for sales and deals related tweets and extract entities from twitter text. Below is an overview of the key findings that were used throughout our development process to help shape our final deliverable.

eCommerce Search Terms

- A research conducted on users in the age group of 13 to 24 resulted in information about sharing trends among various social networking site users. [1]

- 10 most popular activities on Social networking sites: Looking at profile, Updating personal profile, searching for someone, emailing someone, writing on someone else’s profile, reading blogs, listening to music, requesting friendship and looking up someone’s status. Of these 2 activities, Updating personal profile and Looking up someone’s profile are two important activities that aid in gathering information about how information about sales/deals is being diffused across users. [1]

- Among Twitter participants, 20% of status updates are specifically sharing information about brands leading us to consider specific stores or brands as a search term during the Tweet extraction. [1][13]

- Organizations use Twitter as part of their marketing strategies: Twitter as response mechanism - complaints or queries (if lower number of tweets, then reach and efficiency is lower).The efficiency can be increased by retweeting other’s tweets and using mentions in tweets. Eg. Microsoft does this, which would steer us towards including Retweets in our analysis [6]

- Price, discount rate, category matter to the user who shares a daily deal. 70% of tweets shared via Twitter are in the range of 40-60% discount. Also deals present at multiple locations of a store are more likely to be shared. These aspects could be included during the Tweet filtering process to gather more specific data. [9]

Geo-Location Techniques

Measuring geographical regularity of local crowd behaviors by plotting the pattern of these 3 main indicators.

a. How many tweets are posted?
   #Tweets: the number of tweets that were written in an RoI within a specific period of time.
b. How many users are there?
   #Crowd: the number of Twitter users found in an RoI within a specific time period.

c. How active are the movements of the local crowd?
#MovCrowd: the number of moving users related to an RoI within a specified period of time.

The measure of the number of Tweets at a geographical area along with determining the normal crowd behaviour would be beneficial in detecting an event. [2]

- Much less % of the microblog documents are geotagged. Hence determining a method to identify the location on non geotagged documents which is proven to increase the result by 115 times. [5]

- Challenges with geotagging in Twitter:

a. Error rate of 967km
b. Since Twitter documents are short, it may be impossible to predict the location solely based on terms

Alternative solution: Filtering method to detect documents with common theme allocated to the same place. [5]

- Geotag Allocator:

a. Create a database of place names
b. Place names are extracted from Twitter
c. Compare both to find a similarity. Smaller the variance, more precise the name is
d. If there is large variance (Eg: McDonalds is very ambiguous). Such entries can be eliminated from database

This could help determining if Tweets are sales based form a specific geographical area even though they may not be Geotagged. [5]

- Using Geolocation in Tweets to analyze large groups of people. Analyzing 2 types of tweets to gather geolocation data: (1) specifically geo-tagged tweets and (2) text only tweets where they had to extract the lat/long (geo-code) from the tweet by translating the place names using Google's geo-naming service. This could help determining if Tweets are sales based form a specific geographical area even though they may not be Geotagged. [7]

Tweet Analysis

- User/tweet categorizations (i.e. by gender, by geographic region, by tweeting habits, etc.) can be determined by taking a look at "trending" topics via the #hashtag in the tweets. UID as a measure of time as opposed to the datetime of the tweet. Could be useful to note as a way of filtering for tweets within a certain time period. The inclusion of Hashtags as a Tweet filter measure would be profitable while gathering sales related tweets [3]

- The effectiveness of Twitter data as a source of BI systems may critically rely on how well structural information on Twitter is exploited and how novel text mining techniques can be applied to analyze tweets.

1. Weak ties are more likely to lead to retweeting
2. For an advertising tweet, check if it contains an url
3. Use a dictionary lexicon to further filter out the tweets
4. 4 Categories of tweets helps: intention, positive, negative and neutral

Could consider if Url linking to specific deals could be included in the filter algorithm.[4]

- Metrics designed to capture text based opinion divergence in product reviews, adventure into an unexplored frontier in sentiment analysis. Their impact on consumer purchase behavior and product sales has significant

implications for Word-Of-Mouth driven marketing research. [8]

- Among Twitter participants, 20% of status updates are specifically sharing information about brands leading us to consider specific stores or brands as a search term during Tweet extraction. [9]

- Traditional NLP tools do not work well with twitter data. Hence, utilize a named entity tagger trained on in-domain Twitter data presented in previous work. First annotate a corpus of tweets, which is then used to train sequence models to extract events. [10]

- There are three factors to look for when trying to extract events and event descriptions from Twitter. [12]

1. Main entity extraction: Use a large corpus and see manually the weightage of each word. If the inverse frequency is higher than other words, then it assumes higher importance and thus is the main entity. From than we can take the inverse document frequency.
2. Extracting actions and opinions: entity followed by a verb or noun phrase
3. Audience opinion extraction: Verbs preceded by a main entity or a pronoun followed by a verb phrase and then the main entity.

- Much of words in tweets are "ill-formed." For example, one can write "tomorrow" as "2morrow." There is a certain method that can be used to transform an "ill-formed" word (T) to a single "standard formed" word (S). There are 3 steps to this process: [11]

1. Identify which words are out-of-vocabulary (OOV). We can use GNU Aspell Dictionary to identify OOV's. The categories of OOV's include: Letter&Number, Letter, Number Substitution, Slang and Other [11].
2. Identify which words are ill-formed in relation to the string context. There are several methods where we can extract bi-grams to establish context. [11]
3. Candidate selection where we select an S for each T. There are a variety of algorithms that can be used for this.

This approach of lexical normalization may be cumbersome for our purposes since we are searching for only a few terms and are not in a need to normalize a large volume of tweets. The techniques used in this paper can be helpful for identifying OOV's.

Bibliography

1. Park Y., Jaimie, and Chung, Chin-Wan. "When daily services meet Twitter: understanding Twitter as a daily deal marketing platform." *WebSci'12 3rd Annual ACM Web Science Conference.* Pages 233-242. 2012. http://dl.acm.org/citation.cfm?id=2380748&bnc=1

2. Burton, Suzan et al. "Interactive or Reactive? Marketing with Twitter." *Emerald Group Publishing Limited.* 2011. http://www.emeraldinsight.com/case_studies.htm/case_studies.htm?articleid=17003316&show=html

3. Cheong, Marc, and Lee, Vincent. "Integrating web-based intelligence retrieval and decision-making from the twitter trends knowledge base." *SWSM'09 2nd ACM Workshop on Social Web Search and Mining*. Pages 1-8. 2009. http://dl.acm.org/citation.cfm?id=1651439&bnc=1

4. Rui, Huaxia, and Whinston, Andrew. "Designing a social-broadcasting-based business intelligence system." *ACM Transactions on Management Information Systems.* Volume 2 Issue 4. 2009. http://dl.acm.org/citation.cfm?id=2070713&bnc=1

5. Watanabe, Kazufumi, et al. "Jasmine: a real time local-event detection system based on geolocation information propagated to microblogs." *CIKM'11 20th ACM International Conference on Information and Knowledge Management.* Pages 2541-2544. 2011. http://dl.acm.org/citation.cfm?id=2064014&bnc=1

6. Lee, Ryong, and Sumiya, Kazutoshi. "Measuring geographical regularities of crowd behavior for Twitter-based geo-social event detection." *LBSN'10 ACM SIGSPATIAL International Workshop on Location Based Social Networks.* Pages 1-10. 2010. http://dl.acm.org/citation.cfm?id=1867699.1867701&coll=DL&dl=ACM

7. Wakamiya, Shoko, et al. "Crowd-sourced urban life monitoring: urban area characterization based crowd behavior patterns from Twitter." *ICUIMC'12 6th International Conference on Ubiquitous Information Management and Communication.* Article No 26. 2012. http://dl.acm.org/citation.cfm?id=2184784&bnc=1

8. Zhang, Zhu, et al. "Deciphering word-of-mouth in social media: Text-based metrics of consumer reviews." *ACM Transactions on Management Information Systems.* Volume 3 Issue 1. 2012. http://dl.acm.org/citation.cfm?id=2151163.2151168&coll=DL&dl=ACM

9. Jansen, Bernard, et al. "Being networked and being engaged: the impact of social networking on ecommerce information behavior." *iConference'11.* Pages 130-136. 2011. http://dl.acm.org/citation.cfm?id=1940779&bnc=1

10. Ritter, Alan, et al. "Open domain event extraction from twitter." *KDD'12 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining.* Pages 1104-1112. 2012. http://dl.acm.org/citation.cfm?id=2339704&bnc=1

11. Han, Bo, and Baldwin, Timothy. "Lexical normalization of short text messages: makn sens a #twitter." *HLT'11 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies.* Pages 368-378. 2011. http://dl.acm.org/citation.cfm?id=2002520&bnc=1

12. Popescu, Ana-Maria, et al. "Extracting events and event descriptions from Twitter." *WWW'12 20th International Conference Companion on World Wide Web.* Pages 105-106. 2011. http://dl.acm.org/citation.cfm?id=1963246&bnc=1