1 Introduction

Twitter is a social micro-blogging site where users can submit short messages (limited to 140 characters) via a web interface, instant message, or text message and are urged to post messages about what it is they are currently doing. While most users form small communities and specifically follow the messages of people they know, Twitter also has a public timeline which just a running list of all recently added messages from most users.

TwitterVis was conceived to be a tool to explore the vast corpus of Twitter messages spontaneously and discover new patterns. Some key questions we sought to answer were: are there geographic hot spots where people use twitter, is there a temporal relationship to twitter messages and geography, can we discern meaning from the individual twitter messages by looking at the corpus as a whole, and can we then discern meaning by looking at the messages from a specific area and time?

2 Methods

We populated our database of recent Twitter messages by continuously polling the public timeline over Twitter’s API. A python script was written to access the XML feed that Twitter provides and to extract data from it. This data included the message id, the message itself, the time the message was created, the user id, user name, the user’s screen name and the user’s location. The message text and the user’s location information are input by the user as free form text. With regard to the location field this meant that some user’s may have left the location field blank or provided imprecise information or information in a variety of formats (such as "San Francisco", "San Francisco, CA", "SF, CA", or even just "CA"). This made extracting meaningful location data difficult, but we found that most users input location information that could be meaningfully parsed. In order to geocode the users (provide a latitude and longitude for their location) we passed the user location data to the Yahoo Geocoding Service and updated the user table with the resulting latitude and longitude information.

We are storing all information in a MySQL 5.01 database and currently have about 700,000 unique messages from 70,000 users. Message and user data are stored in separate tables. Message data is stored in a single table with the unique message id serving as the primary key along with a separate column for the user id to allow joins against the user table where the user id serves as the primary key. In order to extract information from our database we created a RESTful API using PHP. Requests are encoded with URL parameters and the API returns XML documents with the data requested.

2.1 Geographic Track

One purpose of TwitterVis is to facilitate spontaneous exploration of Twitter content and trends based on regions. Did hot spots exist where Twitter usage was particularly prevalent? Was there a temporal aspect to Twitter usage more frequent during certain times of day? And did different geographic areas have specific memes within the messages; that is to say, were people in the same geographic location Twittering about the same thing?

In order to solve these problems we decided to implement a dynamic query interface which would display the most frequent terms of the most recent messages in a particular region. To accomplish this we set about on an interface that would provide the user with the ability to select a specific geographic location via a map interface. Geographic interfaces have become considerably more prevalent in recent years as the Google Map API and Yahoo Maps Web Services functionalities have become mainstream. Each of these services allows developers to use their map interface to encode geographically relevant information. For our
project we used a similar map functionality from Modest Maps, which is an open source and lightweight map display library for Flash.

The map features a selection radius around the center of the map panel. Upon panning or zooming, the map is redrawn and the most recent messages within the region are retrieved using the REST API we developed. A cloud of red dots within the circle indicates the locations of the authors of the messages. The dots are translucent; a more saturated circle indicates many messages originating from the same location. The most frequent terms are computed and displayed alongside the map in descending order; the most frequent terms would appear first.

Ultimately the aim of the geographic exploration tool is to allow users to get a quick idea of what's currently being talked about in a particular region using an interface that they already find comfortable and intuitive.

2.2 Term Frequency Track

2.2.1. The problem: Twitter has recently introduced track functionality. Users can use this functionality to track all messages containing keywords of their interest. For example, if a user is interested to know what people are talking about 'radiohead', he or she types in 'track radiohead' using IM interface. All subsequent tweets on public timeline will be tracked for 'radiohead' and the user will be notified of any messages that have the keyword in them. If the user chooses to track high volume, popular keywords like google, he or she is notified of a large number of tweets every minute. This can be both a distraction as well as too much information to handle. There is no 'offline' mechanism though which user can look at all the messages at some later point of time. Also, it can be really hard for the user to read every single tweet if there are large numbers coming in every minute. There may be repetitions in the notified messages. Some of the messages which contain the keyword may be talking about aspects which may not be interesting to the user. This gains significance especially if the volume of tweets goes up beyond what is manageable. We wanted to build a tool that help users manage this information overload through a visualization that lets him or her explore the messages meaningfully.

2.2.2 Related Work: Popular community-voted news website Digg has built a visualization tool that displays stacks of diggs a news story has received. The visualization displays a snapshot of digg's time line from the moment you start the visualization. It continues to build stacks as people digg the story real time. It uses hue to indicate the number of diggs received by the story before the start of the current screenshot. The visualization animates tetris like falling blocks onto the stacks to indicate a digg by a user. The bottom of the page is used to display the headline of the actual news story as the block falls down. These headlines are displayed in a font whose size is weighed by the number of the diggs the story received.

Assuming a track word instead of a news story, our twitter track overload problem can use many of the features of this solution, but many of the problems remain unsolved. User still has to read the messages real time before they
disappear into the hue of each stack. The only real advantage user gains is an idea of the volume of messages for each track word. We propose an modified solution based on the properties of the contents of a tweet.

2.2.3 Proposed approach: Tweets have the property being short: 140 characters at the maximum. They usually have a well formed message, although they are not always well typed out. They are relatively easy to read compared to other longer forms of web text like blogs and news stories. Since they are usually personal in nature, there is little similarity between any two tweets. Shorter length has other implications. Since its easy to create them, higher volume of tweets can be quite high.

These properties have certain important change implications to the Digg Labs design of stack visualization. Tweets can be stacked into different bars based on track words user is interested in. Since there is little similarity between different tweets stacked on the same bar, it is important to allow the user to further drill down the volume of the stack to see what messages are in. Since popular keywords like google may have a few hundred tweets per day, it can be hard to read every single message that contains the keyword. Instead, we calculate term frequency for all the messages in the stack and produce a tag cloud based on relative frequency of different words used in this stack of messages. Clicking on each tag in the tag cloud lists all the messages that highlight both the original keyword as well as the tag that was just clicked.

2.2.4 Text Processing: A two-day snapshot of public time line was used for the purpose of visualization. Using available API, an XML file has been generated which contains messageID, time stamp, name of the user who created the message and the actual message for every message. After stripping out message IDs, time stamps and username and other XML tags. After stripping off XML tags from the snapshot, we use basic text processing techniques to generate a tag cloud.

We start off creating a list of stop words. We used the standard stop word list created by the Computer Science department at University of Glasgow. The list consists of articles (a, an, the), common prepositions (like of, to, in, for, on, with, as, by, at, from) etc. The standard list also consists of words like now, today, tomorrow. These words are special of significance in the context of twitter for an important reason. A study by Akshaya Java et.al. has found that most users use twitter for daily chatter. Understandably words like now, just, today and tomorrow occur frequently in such daily chat. So we have removed all such words, that can potentially be of significance in the context of twitter, from the standard list of keywords.

The next step we did was make a list of stop phrases. By stop phrases, we mean phrases that are not made of stop words but at the same time appear a lot as part of people's messages. Making such a list was entirely based on observation of random messages and their related messages. An insight we gained however is the fact that, messages with stop phrases were highly dependent on the user generating these messages. i.e. Tweets by same user tended to have often repeated phrases. It is probably a good idea to deal with this problem by using tf-idf and term weighting techniques than manually constructed lists of stop words. But in the interest of time, we had to live with a manually constructed list of stop phrases.

After deleting stop words and stop phrases from the list of original tweets, we made a copy of all tweets which would eventually be our corpus for finding track words and making tag clouds. For the purpose of finding term frequency, we wanted to further clean up the corpus by removing URLs. We use aggressive regular expressions in Python to strip off the URLs from the corpus. We noticed that the corpus is littered with HTML entity characters like &#39; and &#39; and & in this example. We use standard HTML entity conversion table for lookup. We finally strip off any punctuation like period not followed by a whitespace. The corpus is now ready of for tokenization. We tokenize the corpus and pass the tokens through a Porter stemmer.

Alternatively, we wanted to make a second variant of the corpus that separates URLs from text and which maintains a mapping between them. Such an alternate corpus can be used for a digg-style visualization tool that shows new URLs posted for a given keyword. Again, in the interest of time, we had to give up this part.

Ideally, user should be able to enter a list of track words he is interested-in using a text box or other input interface. But for the purpose of demonstration of the visualization technique we used a list of sample track words in a configuration file. We use the stemmed corpus for grep-ing the stemmed track words. A count
of number of messages containing the track word is stored as a dictionary object. A list of all messages containing the track word are passed to a tag cloud generator. The tag cloud generator calculates the term frequency of the tokens and then passes back a HTML snippet of tags with font-weights according to the frequency. Each tag in the cloud links to a page that displays all messages that contain both the track word as well as the tag.

2.2.5 Visualization: The visualization uses three frames, one frame for upper half and two frames split vertically for upper half. The top frame displays a bar graph of track word message volume count. The track bars are arranged in a chronological order in which user entered the track words. While this is one approach, a better approach may be to group track words based on similarity. This can further facilitates making inferences from relative volume for the user. Chronological sequence, on the other hand, helps user quickly know what is happening with the most recent track words. All the track bars are clickable.

When the user clicks one of the track bars, the bottom left frame refreshes with a tag cloud for that particular track word. The tag cloud, naturally, ignores the original track word as it would have dominated the cloud anyways. The original track word is shown on top of the cloud as a reference to whose cloud it is. Clicking on one of the tags refreshes the bottom right frame with all the messages that contain both the track word and the clicked tag. Both the track word and clicked tag are shown in red to help user quickly scan through this relatively tiny subset of messages quickly.

3 Discussion

3.1 Geographic Track

We created a working prototype application using Flash which would query the REST API and display the results on a map interface. The map has both pan and zoom functionalities to let people explore the twitter message corpus we have created. The prototype, however, is not perfect. We were forced to restrict the radius from which messages would be retrieved in order to account for some performance issues with retrieving and loading a lot of message data for a large area. We were also forced, for performance reasons, to limit the number of messages retrieved and so adjusted the API to return only the most recent 200 messages. We also discovered a further limitation while geocoding the users. Twitter has users from all over the world and stores all user input in UTF-8 format, which means there are a great number of non-western character sets. We found these to largely be impossible to parse meaningfully. Because of this the most complete and accurate results will occur in North America and other countries where English is the dominant language.

After using our tool we discovered some broad trends. As could be expected the highest density of Twitter users are congregated in large metropolitan areas, but more rural areas are, by no means, unrepresented. By
examining the east and west coasts of the United States the density of users in New York, Boston, Miami, Los Angeles, San Francisco, and Seattle and Vancouver, CA is easily discerned. Moving east to west, however, across the center of the United States shows some expected high density areas such as Chicago, Kansas City, Denver, but also shows a broad smattering of low density messages scattered throughout the rest of the country. This implies that Twitter usage may be more evenly distributed geographically with respect to population density than first thought, but that is a question for future study.

Message content definitely differs between geographic locations. While our current implementation only allows for a cursory overview of the most commonly occurring terms, it does provide limited insight into the nature of the messages. Certain terms occur with relatively frequency across all locations in the US. "Christmas" for example, appears in the list of frequent terms for almost every location surveyed. Other terms, however, seem to have particular prominence in some locations more than in others. This can partly be explained by the nature of our prototype itself. Because our application only retrieves a snapshot of the 200 most recent messages for a given location there will be a certain level of distortion between locations due to time zone differences. As an example, one could expect a relative frequency of the term "dinner" at 7 PM in Los Angeles, but might see the term "drinks" with a higher frequency in New York at the same time, due to the fact that New York is three hours ahead of Los Angeles.

Ultimately this type of time zone distortion could be accounted for to a certain extent programmatically. This led us to consider implementing a time slider (a graphical timeline that would allow a user to pick ranges for which data should be gathered) into the user interface which would allow a user to select a date and time range to within an hour. This would provide the user with a greater ability to explore the corpus along multiple dimensions and allow a user to gain further insight into the messages. And for all but the most recent messages it would allow a user to compare the terms between two locations for a given period of time, a full 24 hours for example, or a specific time frame per location adjusted for time zones; such as what are people in Los Angeles and New York twittering about at 2 PM local time on a Sunday? This time slider could also be used to isolate potentially significant moments. It would be possible to see what people are twittering about on major holidays or during times of events that affect a broad group of people simultaneously (such as natural disasters, major social events such as protests or rallies, or even large sporting events like the Olympics).

3.2 Term Frequency Track

The tool greatly reduces the complexity of scanning through long list of messages by breaking them into automated categories. To achieve this purpose, it uses visual aides like interactive graphs, tag clouds and highlighted words. The tool can be really helpful in gaining insights about people's public tweeting behavior. Generic track words like today, tomorrow create an atmosphere of community thriving with activity. Track words like NYT or BBC tell you what news is hot or cool. Track words like live, now and watching tell you what people or watching on TV. If this interface was mashed up with the other track that features maps, users would be able to know if there is anything happening now in their neighborhood. Such track words also require alternate interfaces to account for real time nature of the messages as they would be of far lesser value once they expire.

4 Future Work

It greatly benefits the users to partially integrate both tracks. Geographical locations of tracked messages can help users gain insights into happenings in their local neighborhood. Other future possibilities for term frequency track have been mentioned in its main section. They include, creating an input interface for entering track words, creating a digg like interface for URLs as well as track words like now and live that have a significant real time component in them. Tag clouds can be further improved using word net similarity. Bigram analysis may help to spot trends in music and TV using track words like 'listening to ____' and 'watching ______'.

It would also be useful to explore ways of optimizing query results and processing time as well allowing the user to control how large a dataset they want to explore. This could be accomplished by returning a subset of the total number of results for specific query while also informing the user of the full size of the total result set and letting them page through the
There is also a great deal of information in the database that would most appropriately be visualized through more standard means such as line and bar charts. This would include statistical information about total message counts for a given time frame or the top twitter message contributors. If these types of functionalities could be incorporated into the overall user interface then a user would have a broad selection of tools at their disposal to explore and make sense of the vast amount of Twitter message and user data that is available.

5 References

Google trends: http://www.google.com/trends


1 http://www.twitter.com
2 Users can choose to not make their messages public which will also remove them from the public time line