THE PENNSYLVANIA STATE UNIVERSITY
SCHREYER HONORS COLLEGE

DEPARTMENT OF COMPUTER SCIENCE

TWEET CATEGORIZATION WITH PATTERN DIFFUSION

ZACHARY BUSH
Spring 2012

A thesis
submitted in partial fulfillment
of the requirements
for a baccalaureate degree
in Computer Science
with honors in Computer Science

Reviewed and approved* by the following:

Dr. Meng Su
Associate Professor of Computer Science
Thesis Supervisor

Ronald McCarty
Instructor in Computer Science
Honors Advisor

* Signatures are on file in the Schreyer Honors College.

ABSTRACT

In the last several years, there has been an influx of socially oriented technology. Various websites such as Facebook and Twitter that are driven by users' sharing information have appeared. The field of social computing is still very new, and the information available through these interactions is being explored. This paper proposes applying data mining methods such as PCA (Principal Component Analysis) and Diffusion Geometry to the information that users exchange to efficiently categorize them. Additionally, this paper proposes applying known ranking methods to identify the users that cause topics to become popular.

In order to categorize users' tweets, each tweet message is represented as an attribute vector, which is made by the frequencies of certain keywords in the tweets' content. The top one hundred most frequently appearing meaningful words are picked as the keywords that each tweet would be represented by.

It is impossible to display the clusters of the tweet vectors created by k-means because of the high dimensionality of the data. The various keywords may be related, so the intrinsic dimension of the data set is much smaller than one hundred. Thus, the two methods (PCA and Diffusion Geometry) are used to reduce the dimensionality of the data. Our calculating results on about one thousand real tweet messages show that these two mining methods are promising in their ability to categorize tweets into groups.

TABLE OF CONTENTS

LIST OF FIGURES iii

ACKNOWLEDGEMENTS v

Chapter 1 Introduction 1

Chapter 2 State of Research 2

Chapter 3 System Requirements 5

- 1.0 System Summary 5
- 2.0 System Platform and Environment 5
- 3.0 Interfaces 6
- 4.0 Data Description and Organization 7
- 5.0 Design Considerations - Detail Characteristics 7
- 6.0 Acceptance Criteria - Quantitative 8
- 7.0 Maintenance 11
- 8.0 Security 12
- 9.0 Delivery and Installation 12
- 10.0 Proposed Methods and Procedures 13

Chapter 4 System Architecture 14

- 1.0 Architecture Overview and Scope 14
- 2.0 Tier Definitions 16
- 3.0 Software Module 17
- 4.0 Hardware Architecture Diagram 19
- 5.0 Network Architecture 20
- 6.0 Data Architecture 21
- 7.0 User Interface Architecture 23
- 8.0 Technology 25
- 9.0 Phases 26
- 10.0 Preconditions and Inputs 26

Chapter 5 System Design 27

- 1.0 Component Level Design 27
- 2.0 Data Design 40

Chapter 6 Results and Analysis 42

Chapter 7 References 48

Appendix A - Glossary 50

LIST OF FIGURES

Figure **1** - Architecture Overview. 14

Figure **2** - Tier Definitions. 16

Figure **3** - Software Modules. 17

Figure **4** - Hardware Architecture. 19

Figure **5** - Network Architecture 20

Figure **6** - Data Architecture. 21

Figure **7** - User Interface. 23

Figure **8** - Number of Tweets Retweeted. 23

Figure **9** - Sample User Layout. 24

Figure **10** - UML Diagram. 27

Figure **11** - Master Page UML. 28

Figure **12** - Query Page UML. 28

Figure **13** - VisualizePage UML. 29

Figure **14** - ResultsPage UML 29

Figure **15** - Landing Page UML. 30

Figure **16** - Geocode Class UML. 30

Figure **17** - Query Class UML. 31

Figure **18** - ResultSet Class UML. 33

Figure **19** - AJAX Module UML. 34

Figure **20** - Tweet Retrieval UML. 35

Figure **21** - DataParser UML. 36

Figure **22** - DataProcessor UML. 38

Figure **23** - Data Design 40

Figure **24** - First Three Attributes of Original Data. 44

Figure **25** - PCA Method.....45

Figure **26** - Diffusion Map, Clustered by Original Data.....46

Figure **27** - Diffusion Map, Clustered by Diffusion Map Data.....46

ACKNOWLEDGEMENTS

I am extremely grateful to have been given the opportunity to work with my thesis supervisor, Dr. Meng Su, throughout my undergraduate studies. His guidance, patience, and support even late into the night while working on this research enabled me to explore my interests and the process of research in computer science, which has led to the completion of this honors thesis. I could not have done it without him.

I would like to thank Chuck Burchard for helping me find my interests in my undergraduate studies. His insistence on analyzing problems in a multitude of ways, as well as instilling an experimental style of coding for problem solving helped me numerous times.

Finally, I would like to thank Ron McCarty and Gary Walker for being patient and accommodating when I needed it most. Both of their support helped me to stay motivated even through the toughest parts.

Chapter 1

Introduction

This is a system with the goal of collecting and analyzing data from Twitter. The system will collect messages and meta-data from the individual tweets to analyze specific characteristics. These characteristics will be measured and analyzed to categorize tweets' content and the flow of information through Twitter.

The Tweet Categorization system will be designed to interface with the Twitter API to gather the data. The data gathered will be available to authorized users so that the data can be analyzed. The collection program will be designed to interface with the analysis program so that the majority of the data processing can be made autonomous.

Pattern diffusion will prove to be a more effective method of categorizing user tweets. Similar methods of categorization require training sets, while pattern diffusion does not. Instead of requiring training data, the pattern diffusion method requires sets of keywords or characteristics to analyze as a vector. The effectiveness of pattern diffusion will be tested by rating the accuracy and speed of its method, compared to other categorization methods.

The tweets that create a *trending topic* can be categorized and analyzed to determine which users were vital in causing the topic to trend. The tweets' metadata and contents will both be analyzed to create a network. This will be tested by analyzing the accuracy, consistency, and speed in identifying key users and messages.

Chapter 2

State of Research

Twitter exists for users to share what they are doing on a day to day basis. However, in previous studies, this function was not the primary use of the website. Instead, users are using it to send messages to each other, even though the website is not designed for inter-user communication. (Herring et al, 2009). Users that use Twitter as a method to message other users directly are particularly interesting, as they have much more direct interaction with other users.

A previous study has been done to use the HITS algorithm to find the hubs and authorities within the network. (Java et al). This algorithm was proposed to find which users are the most central to the network of users within the websites. Based on this detection, communication can be identified within these links to find communities, often based on common ground between the users (Java et al). Another study has been done to determine the relation between the number of followers or friends that a user has and the activity of such a user (Huberman et al, 2008). This research found that as a user had more friends or followers (links), the user was more active. This research is relevant to the study done by Java et al, which finds the central users. By researching these "power users", and their interactions with other users in the community, we can build a network through which information is disseminated. On the contrary, there is a "possibility that 'ordinary influencers'--individuals who expert average, or even less-than-average influence-- are under many circumstances more cost-effective."(Bakshy et al 2011) This

study suggests that the source of the content diffused between users may play just as large, or even larger role than the importance of the user himself in determining how information spreads. This observation should be tested further, along with the different types of "cascades", chains of a specific piece of information being passed between users.

A study by Krishnamurthy et al. suggests that there are several distinct groups that can be characterized by analyzing Twitter data (Krishnamurthy et al, 08). This subdivision of users will be useful in breaking down the groups of users more specifically, which in turn will allow blocking of the 'user' variable. If similar users are found, the types of messages that they send out could be comparable. Additionally, if the trend of similar user equals similar message is true, then demographic information can be traced back to the individual that posted the message to begin with.

In testing the importance of the source of information, a topic-sensitive PageRank algorithm could be used. The topic-sensitive algorithm also can apply method of query-sensitive scoring. This method of scoring will enable the results to be filtered based on the context that the query results appear in (Haveliwala, 2002). Combining this technique with others such as pattern diffusion will allow for multiple layers of processing with the data--allowing for more accurate categorization to occur. The adaptive version of the PageRank algorithm will be useful in processing a very large dataset, particularly useful when millions of users are distributing millions of messages very quickly (Kamvar et al.)

There are also different methods of transmitting information via twitter. Three different methods: indegree, retweets, and mentions all can be used to measure the influence of a particular Twitter user. The first, measures popularity, but is not related to

engaging audience. Retweets are measures of the content value, and mentions are measures of the name value (Cha et al, 2010). These three characteristics will play an important role in determining the influence of the user and the messages that he relays to the rest of the network. The type of messages a user tweets can help to determine the extent of the user's influence and the type of influence he has.

Methods for handing this large set of data have been developed, currently there are advances being researched. One of these methods is to use diffusion geometries to categorize data that has a large number of variables. "Diffusion geometries refer to the large-scale geometry of a manifold or graph representing a data set, which is determined by long-time heat flows on the manifold/graph/data set." (Mauro). These same geometries can be applied to the messages based on their content, keywords, and users that interacted with these messages.

Chapter 3

System Requirements

1.0 System Summary

This section specifies the basic system requirements.

1.1 *This system shall categorize Tweets.

The purpose of this categorization will be to organize the tweets by topic and audience, according to the tweets' content, users mentioned in the tweet, and the other tweet metadata.

1.2 *This system shall analyze Twitter trends.

The trends will be analyzed to determine both the users that started the trend and to predict future trends.

1.3 *This system shall interact with Twitter via REST protocols.

The Twitter API provides an interface to gather data through asynchronous function calls to their website. These protocols enable data to be gathered in real time, according to the users requests.

1.4 *This system shall be web-based.

2.0 System Platform and Environment

This section specifies the environment that the system will run in.

2.1 *This system shall be platform independent.

The system will be browser based, so the operating system will be irrelevant.

3.0 Interfaces

This section specifies the ways that this system will be interacted with.

3.1 User Interface

3.1.1 *The UI shall require authentication.

3.1.2 *The user shall be able to view visualizations based upon his query.

These visualizations will show the results of the users' searches in a graph form, so that a visual representation of the web created by user interactions and so that the users can easily see how tweets cluster together based on content.

3.1.3 *The user shall be able to search on custom terms.

3.2 External Interface

3.2.1 *The system shall interact with Twitter services via REST protocols.

3.3 Internal Interface

3.3.1 *The system shall pass data from the web interface to the server.

3.3.2 *The server shall do all the data processing.

4.0 Data Description and Organization

This section specifies the data that will be stored by the system.

4.1 *The system shall store tweet content in a database.

4.2 *The system shall store tweet meta-data in a database.

This metadata will contain fields such as: ID of an existing tweet that the tweet is a reply to, the creation date, the authors 'biography', the location, favorites, language, and other user information.

4.3 *The system shall store user searches.

4.4 The system should dynamically process the request.

If the system is taking too long to dynamically process each request, it would be possible to cache specific results, or queue the analysis for a later time, sending the user a notification when the analysis is complete.

4.5 *The system shall store results of analysis in a database.

The exact size of the tweets, as well as the content to be stored pertaining to each tweet is to be declared, based on the analysis techniques used. Currently, all data pertaining to the tweet will be stored, however, this is subject to revision.

5.0 Design Considerations - Detail Characteristics

This section specifies the detailed information about the system.

5.1 *Web programming shall be used to develop the client side.

5.2 *The system shall use a scripting language to parse the data.

5.3 *The system shall use a server-side language to process the data.

6.0 Acceptance Criteria - Quantitative

This section specifies what is necessary to quantitatively and qualitatively accept the requirements.

6.1 System Summary

6.1.1 *Requirement 1.1 shall be accepted if the twitter data can be categorized by topic and user information.

6.1.2 *Requirement 1.2 shall be accepted if twitter trends can be traced back to the original users. Additionally, it shall be accepted if future trends can be predicted.

6.1.3 *Requirement 1.3 shall be accepted if the system interacts through Twitter using REST protocols.

6.1.4 *Requirement 1.4 shall be accepted if the users interact with the system through a web user interface, which passes data to the server to handle.

6.2 System Platform and Environment

6.2.1 *Requirement 2.1 shall be accepted if the users can access the system on any platform.

6.3 Interfaces

6.3.1 User Interface

6.3.1.1 *Requirement 3.1.1 shall be accepted if the user interface requires authentication to access.

6.3.1.2 *Requirement 3.1.2 shall be accepted if the users are presented visualizations based upon their queries.

6.3.1.3 *Requirement 3.1.3 shall be accepted if the user can create their own queries.

6.3.2 External Interface

6.3.2.1 *Requirement 3.2.1 shall be accepted if the system interacts with Twitter through REST protocols

6.3.3 Internal Interface

6.3.3.1 *Requirement 3.3.1 shall be accepted if the system obtains queries from the client and then processes them with server side code.

6.3.3.2 *Requirement 3.3.2 shall be accepted if the server does all of the data processing.

6.4 Data Description and Organization

6.4.1 *Requirement 4.1 shall be accepted if the tweet data is stored in the database.

6.4.2 *Requirement 4.2 shall be accepted if the tweet meta-data is stored in the database.

6.4.3 *Requirement 4.3 shall be accepted if the user queries are stored in the database.

6.4.4 *Requirement 4.4 shall be accepted if the system can process the users' queries and return results in a reasonable amount of time (< 10 seconds).

6.4.5 *Requirement 4.5 shall be accepted if the system stores query results in the database.

6.5 Design Considerations - Detail Characteristics

6.5.1 *Requirement 5.1 shall be accepted if web languages are used to build the interfaces for this system.

6.5.2 *Requirement 5.2 shall be accepted if a scripting language parses the data for necessary information.

6.5.3 *Requirement 5.3 shall be accepted if a server-side language is used to analyze the data.

6.6 Maintenance

6.6.1 *Requirement 7.1 shall be accepted if the system can be periodically updated as the Twitter API is updated.

6.6.2 *Requirement 7.2 shall be accepted if the system can be periodically updated as new techniques for analysis are updated.

6.7 Security

6.7.1 *Requirement 8.1 shall be accepted if the data requires user login to interact with it.

6.7.2 *Requirement 8.2 shall be accepted if there are multiple levels of access available to users.

6.8 Delivery and Installation

6.8.1 *Requirement 9.1 shall be accepted if the system can be run on any platform.

6.8.2 *Requirement 9.2 shall be accepted if the system uses a database to store necessary information.

6.8.2.1 *Requirement 9.2.1 shall be accepted if the system comes with guides relevant to database configuration.

6.8.3 *Requirement 9.3 shall be accepted if the system has useful help documents readily available.

6.8.4 *Requirement 9.4 shall be accepted if the system comes with a user guide.

6.9 Proposed Methods and Procedures

6.9.1 *Requirement 10.1 shall be accepted if test-driven development techniques are used throughout the design process.

6.9.2 *Requirement 10.2 shall be accepted if the results can be independently verified or reproduced successfully with an acceptable level of accuracy which is TBD.

6.9.3 *Requirement 10.3 shall be accepted if the systems' accuracy can be tested and verified by multiple techniques.

6.9.4 *Requirement 10.4 shall be accepted if the system is managed through subversion software.

6.9.5 *Requirement 10.5 shall be accepted if the system is designed using object oriented methodologies.

7.0 Maintenance

This section specifies the requirements to maintain the software after project completion.

7.1 *The system shall receive periodic updates to interact with Twitter.

7.2 The system should receive updates to analytical techniques.

These updates will be required as new research is done. Other techniques may prove to be more successful and should be incorporated, either alone or in conjunction with the currently implemented techniques.

8.0 Security

This section specifies requirements to ensure only authorized users are able to access the data.

8.1 *The system shall require authentication to access the data.

This requirement is in place to restrict the server from being overloaded at any given time. Only publicly available information will be used, but the analysis will likely be processor intensive.

8.2 The system should offer different levels of access to data.

9.0 Delivery and Installation

This section specifies requirements to install system software.

9.1 *This system shall be platform independent.

9.2 *This system shall use a database to store information.

9.2.1 *The system shall come with a database installation and configuration guide.

9.3 *This system shall have descriptive tutorials and help documents.

9.4 The system should have a user guide to help new users.

10.0 Proposed Methods and Procedures

This section specifies code management procedures.

10.1 *The system shall be developed using test-driven development.

10.2 The system should be rigorously tested for accuracy.

10.3 The system's accuracy should be tested using several other analysis methods.

10.4 *The system shall be managed through subversion.

10.5 *The system shall be designed using object-oriented programming.

Chapter 4

System Architecture

1.0 Architecture Overview and Scope

This section describes the overview of the architecture in the system. It includes the user interface, the server, and client interactions.

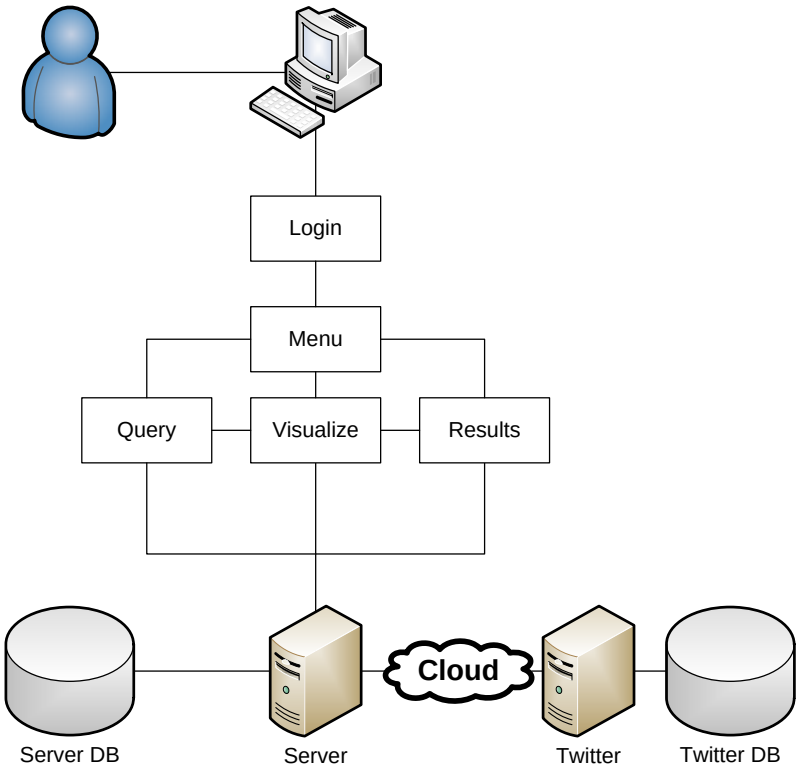


Figure 1 - Architecture Overview

The user will log into the system, where he will be presented with a menu of options: Query, Visualize, and Results. The query option will search Twitter for the most

recent tweets fulfilling the users' parameters. These filters will be on a variety of categories, ranging from demographic information, tweet content, and tweet meta-data, such as time and location. The visualize option will show a visualization for the results that the query returned. These visualizations will display the network between tweets, as well as the relative weight of a particular user's messages. The results panel will display the raw numerical data, as well as the aggregate totals. Additionally, it should be able to review past queries. The server will need to request data from Twitter via the Cloud.

i.e. Suppose a user wants to find and analyze tweets regarding the Occupy Wall Street Movement. First, the user must log in. After the user logs in, he will be presented with the three options: query, visualize, and results. At this point, the user does not have any searches saved, so he clicks query. The options for query appear, including filters. In this example, the user wishes to compare the flow of information between users, based on age, younger or older than 25. The user would select options to create a data set with the two filters: "Age < 25" "#OWS, #OccupyWallStreet". He would then do another query changing the age filter to "Age > 25". Once the query completes, he could look at the network of tweets in the visualize option, or perhaps look at the raw numerical data in the results option. At a later point, the user should be able to retrieve this same search in the results section, without having to recreate the queries.

2.0 Tier Definitions

This section further details the specific layers of the system.

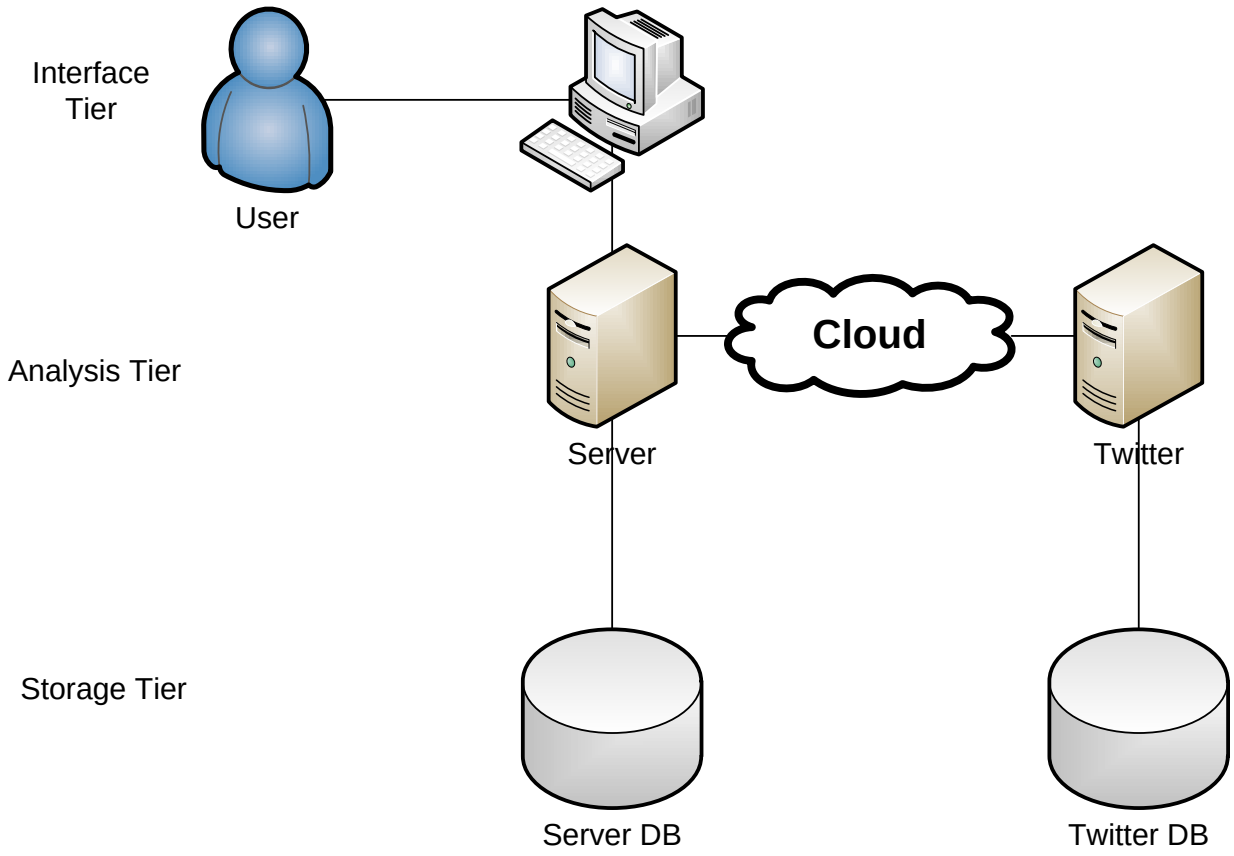


Figure 2 - Tier Definitions

The system will be layered into three main tiers. The interface tier will be where all user interaction takes place. The user will create queries, view stored queries, analyze results, and view visualizations at this tier. The analysis tier will handle the data gathering, parsing, and analysis. This tier will be responsible for taking the data that the user tier queried and preparing it for analysis. After preparation, it will be analyzed at the same level. Finally, the storage tier is where results will be stored. Results can be pulled from in order to gather more data, as well as to retrieve past results.

3.0 Software Module

The software module diagram details the interactions between the various modules in the system.

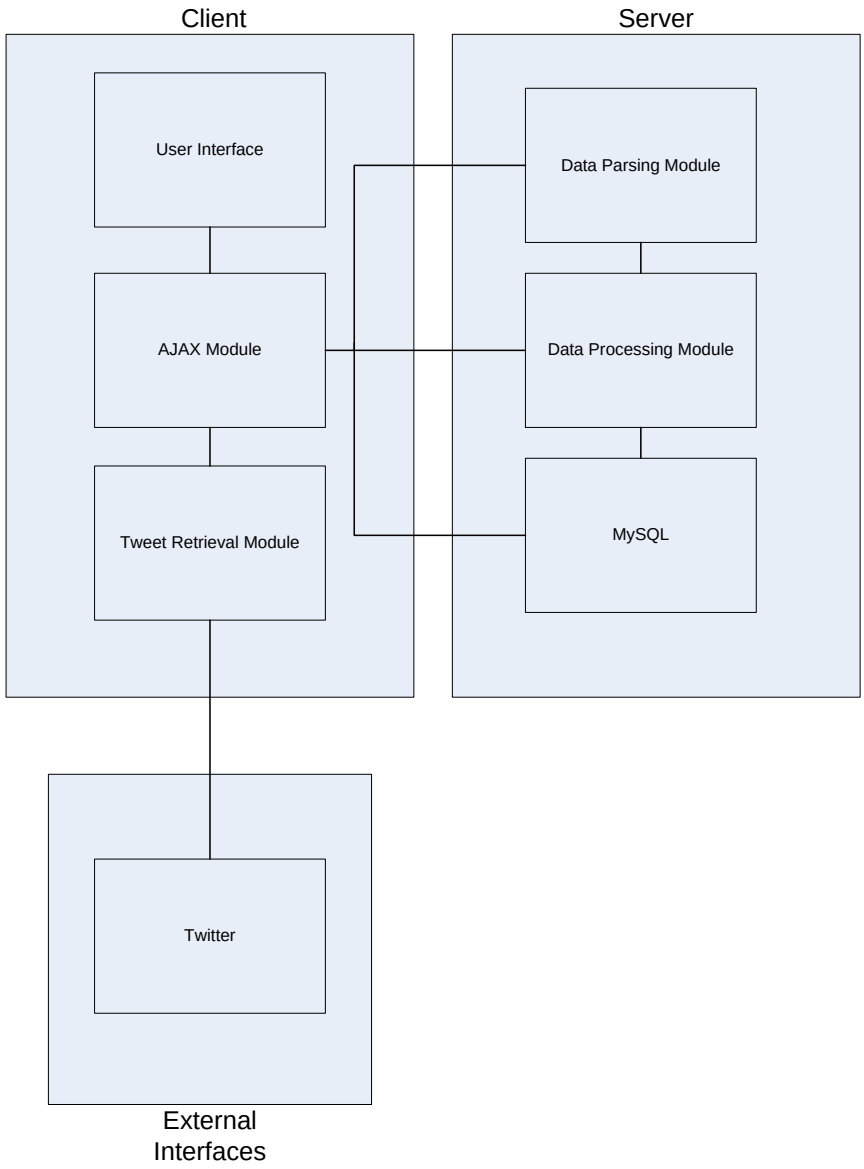


Figure 3 - Software Modules

The user interface will be the main gateway for the user's interaction with the rest of the system. The user interface will pass the user's desired action to the AJAX module. This module will interact with the tweet retrieval module to gather data from Twitter, which is represented as an external source. Additionally, it will be the central point through which data is sent to and retrieved from all modules on the server. The data parsing module will take the gathered data and separate it to its useful pieces. The data processing module will do the necessary calculations on the data gathered. The results will be stored in a mySQL database, which will be accessible via the AJAX module. All connection lines in Figure 3 are bidirectional.

4.0 Hardware Architecture Diagram

This section details the physical hardware in this system.

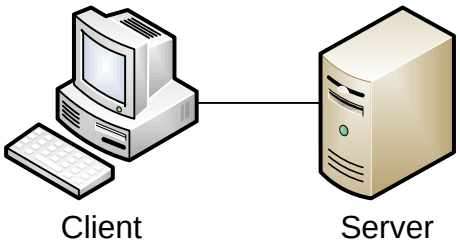


Figure 4 - Hardware Architecture

The system is rather simple, where the client will connect to the Linux server through a web interface. Standard HTML and AJAX will be used to create a dynamic interface between the two.

5.0 Network Architecture

This section details how the client will communicate with the server.

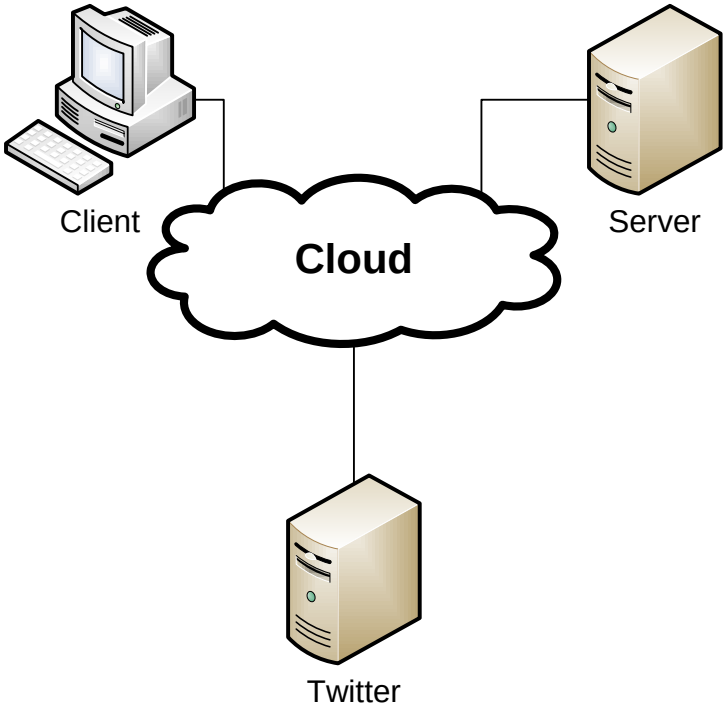


Figure 5 - Network Architecture

The client will interact with the system using HTTP. The server will communicate with Twitter via REST protocols, implemented with AJAX. REST protocols are simply methods of passing information between websites. Typically, asynchronous javascript and xml, a mark up language, are used, but this project will implement JSON (javascript object notation) instead. This will allow for easy interaction with the rest of the javascript modules.

6.0 Data Architecture

This section details the structure of the data gathered.

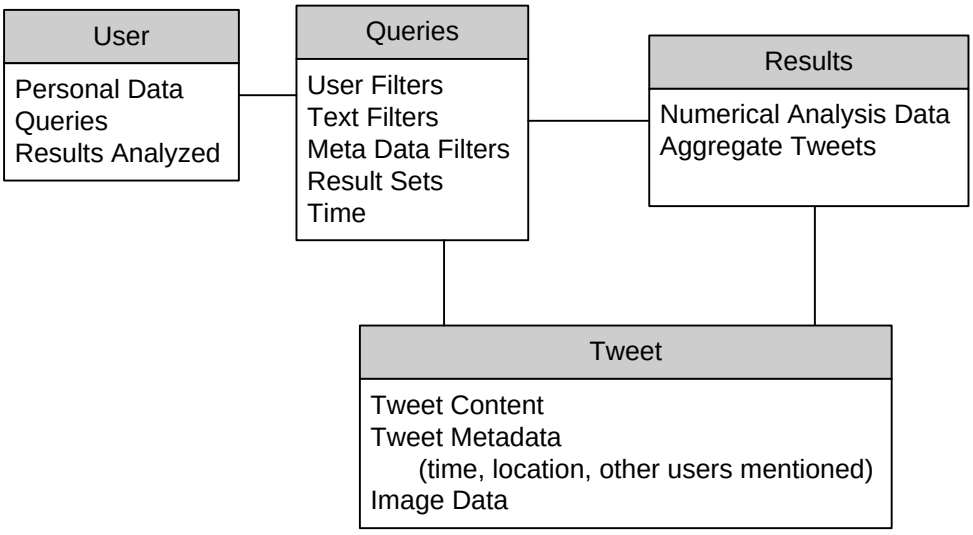


Figure 6 - Data Architecture

Specific user information will be recorded for demographics and login purposes. Queries will be recorded, including the filters that the users chose and the result sets that apply to relevant queries. "User Filters" describes filters about the Twitter users' demographics. In the example given in Section 4.0, an example user filter may be age. Text filters are filters about specific content, such as words or phrases. A text filter may be searching for the word "Occupy". Meta data filters are set to only retrieve results that satisfy certain external criteria such as date and location. A meta data filter may be "Location: New York, NY. or Date: November 5, 2011" The results will contain the numerical data that is a result of the algorithm, as well as the tweets that make up a result set. A tweet will contain its content, as well as metadata such as time, location, other users, retweet status, and hashtags. Additionally, any image data included in the tweet will be recorded. This image data will only be analyzed superficially to see if the same

file is being retweeted. Future development may be able to make use of tools to track changes to the image or similarities, but that is outside the scope of the current project.

7.0 User Interface Architecture

This section details the user interface.

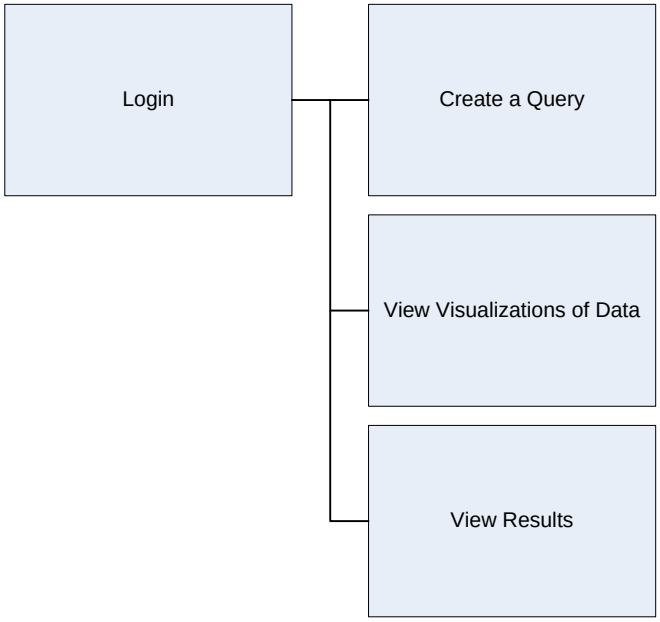
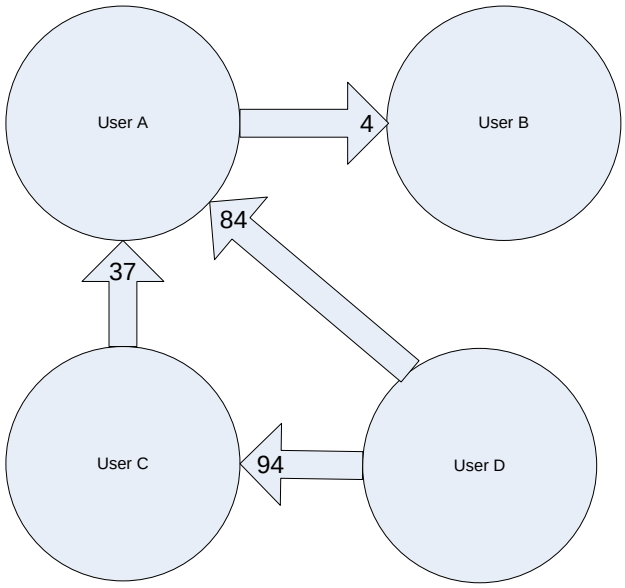


Figure 7 - User Interface



Number of Tweets Retweeted

Figure 8 - Number of Tweets Retweeted

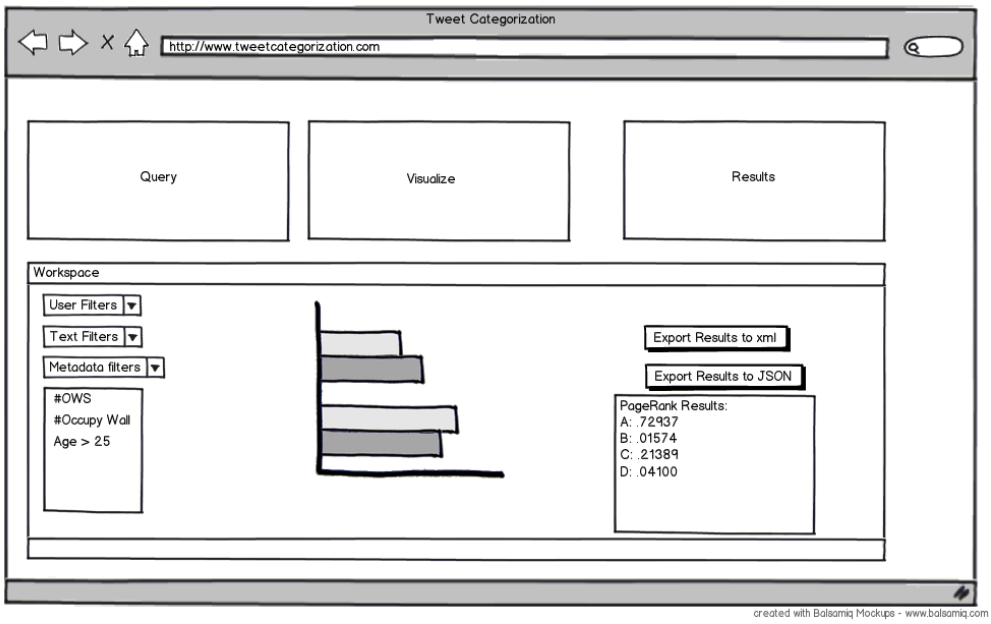


Figure 9 - Sample User Layout

The user interface will also be relatively simple, Figure 7 describes the basic flow. The user must first login to access the rest the system. Once the user logs in, he will be presented with three choices. The first, create a query, will guide the user though the possible options and filters he may place on the data. The second, view visualizations of data, will allow the user to select a record set and view graphs, networks, and information flow diagrams based upon the record set. The last, view numerical data, will present the analysis in such a way that is easy to interact with mathematically.

Figure 8 describes potential visualization. Suppose the number of retweets is the focus of analysis. In this example, we see that User D was the most influential user, as 94 of his messages were retweeted by User C, and 84 of his messages were retweeted by user A. User C is somewhat important, as 37 of his messages were retweeted. User A barely influences this network, and User B does not influence it at all.

Figure 9 describes the basics of the UI. The user will have three major panels, that are easily accessible to perform actions. Underneath, a workspace are will open up. The user should be able to make filters, as well as select from premade ones. The visualization section will provide graphs. Finally, the results section will primarily be used to export for ease of use in another program or AI analysis, but the numerical data will also be displayed.

8.0 Technology

The web interface module will be written with a combination of the following languages: HTML, CSS, PHP, Perl, Python, and Javascript (AJAX included). The database used will be mySQL. The server side parsing and processing will be completed using a combination of Perl, Python, Matlab, and C++.

9.0 Phases

The project will be developed in the following phases. Each phase will have a three step loop, creation of unit tests, coding, and debugging.

1. Installation and configuration of the server and database
2. Website development
3. Interaction with Twitter development
4. Parsing data development
5. Processing data development

10.0 Preconditions and Inputs

This system will require a browser to connect to the interface for the client. The server will require the necessary stack to provide a website, as well as the processing power to manage the analysis.

Chapter 5

System Design

1.0 Component Level Design

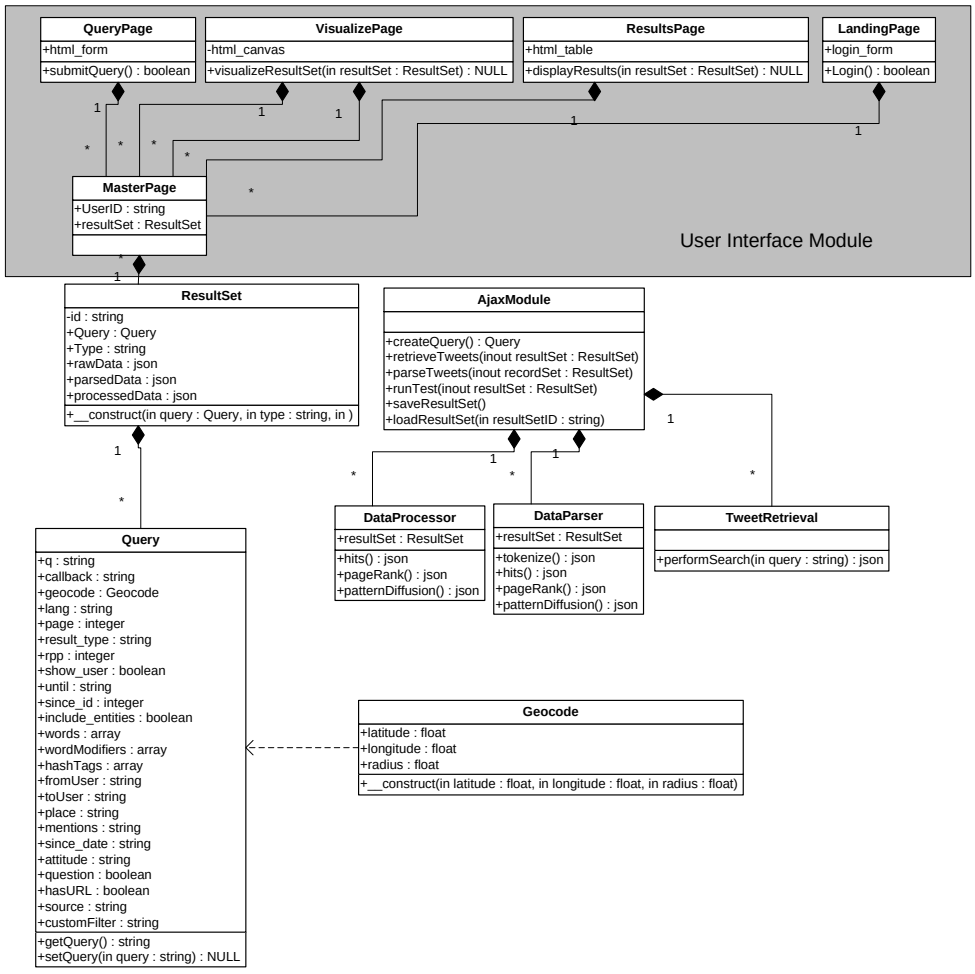


Figure 10 - UML Diagram

1.1 User Interface Module

The user interface module is made up of the five following submodules: MasterPage, QueryPage, VisualizePage, ResultsPage, and LandingPage. Each of the submodules is relatively small, with most of the work done in other modules.

1.1.1 MasterPage

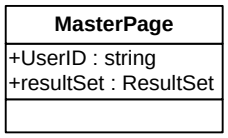


Figure 11 - Master Page UML

Purpose:

This module provides variables which will be available on every page. Both the userID and the result set will be maintained as session variables.

Attributes:

UserID *string*

Holds a unique identifier so that the users activity can be tracked throughout the system. Users actions, queries, and data will be recorded with the userID.

resultSet *ResultSet*

Holds the current ResultSet that the user is interacting with.

Methods: None

1.1.2 QueryPage

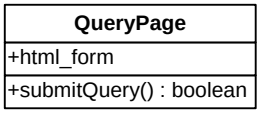


Figure 12 - Query Page UML

Purpose:

This module allows the user to fill out an html form which will create a query.

Attributes:

html_form *form*

This form will hold the data necessary to create a query object. This query object will be used to create a resultSet, which can then be used for data parsing and processing.

Methods:

submitQuery *boolean*

Purpose: To submit the form information in order to create a query object.

Precondition: The form must be filled out.

Postcondition: There should be a query object. Submit query will return true if creation was successful.
Parameters: None

1.1.3 VisualizePage

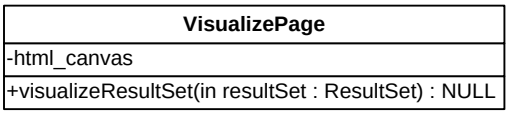


Figure 13 - VisualizePage UML

Purpose:

This module provides the visualizations for the result set.

Attributes:

html_canvas canvas

An html5 canvas object that can be drawn on via WebGl.

Methods:

visualizeResultSet NULL

Purpose: To create an appropriate visualization based on the ResultSet's type and data.

Precondition: There must be a ResultSet.

Postcondition: A visualization should be displayed on the canvas.

Parameters: resultSet a ResultSet

1.1.4 ResultsPage

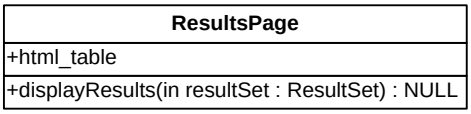


Figure 14 - ResultsPage UML

Purpose:

This module is used for displaying tabular data.

Attributes:

html_table table

The table is used to display tabular data.

Methods:

displayResults NULL

Purpose: To fill the html_data with data from the resultSet based on the ResultSet's type and data.

Precondition: There must be a ResultSet.

Postcondition: The html_table should be filled with data.

Parameters: resultSet a resultSet

1.1.5 LandingPage

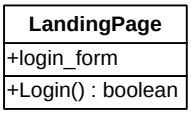


Figure 15 - Landing Page UML

Purpose:

This module is used to enable the users to log into the system.

Attributes:

login_form form

The login form contains fields for a userID and password, which the authenticate the user. The user password will be a standard alphanumeric string determined upon registration.

Methods:

Login boolean

Purpose: To validate the user's login credentials.

Precondition: The form must be filled out.

Postcondition: The user should be authenticated or prompted to try again.

Parameters: None

1.2 Geocode Class

This class contains location data. It has no subclasses.

1.2.1 Geocode Class

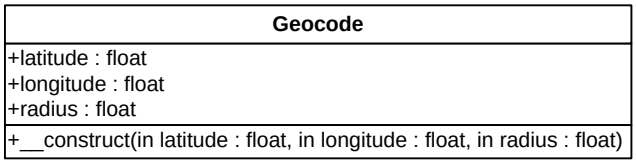


Figure 16 - Geocode Class UML

Purpose:

This module serves to retain location data.

Attributes:

latitude float

Records the latitude corresponding to a location.

longitude float

Records the longitude corresponding to a location.

radius float

Records the radius around a latitude/longitude pair that defines an area.

Methods:

__construct none

Purpose: The Geocode constructor

Precondition: There must be a latitude, longitude, and radius within the appropriate domain. Radius must be greater than or equal to zero.
Postcondition: There should be a Geocode object.
Parameters: latitude — a latitude coordinate
longitude — a longitude coordinate
radius — the radius around the latitude/longitude coordinate that corresponds to a location

1.3 Query Class

This class describes a query. The Geocode class is used within the Query Class.

1.3.1 Query Class

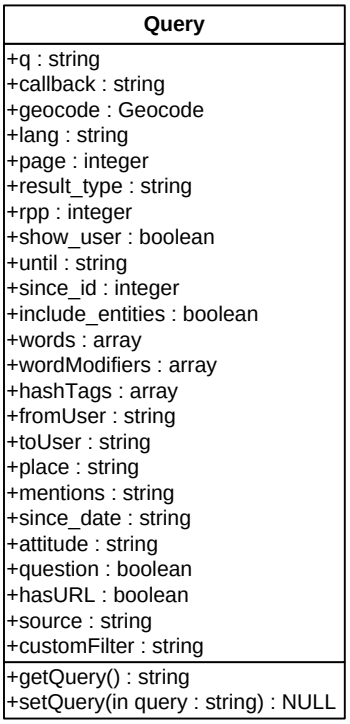


Figure 17 - Query Class UML

Purpose:

This module is used to represent a Twitter query.

Attributes:

q *string*
The query to be sent to Twitter.
callback *string*
A javascript function to be called after the query is returned.
geocode *Geocode*
The location to filter by.
lang *string*

The language to filter records by. Currently, only English will be supported, but there is potential for translators to be implemented.
page *integer*
The page number to return.
result_type *string*
One of the following: mixed, recent, or popular. Filters tweets to only show popular, real-time, or both results.
rpp *integer*
The number of tweets to return per page.
show_user *boolean*
If true, prepends ":" to the beginning of the tweet.
until *string*
Filters tweets by returning tweets up to a specific date.
since_id *integer*
Filters tweets by only showing those made after a specific tweet was made.
include_entities *boolean*
Determines whether or not to return the tweet metadata.
words *array*
An array of words to search for in Tweet content.
wordModifiers *array*
An array of modifiers to the words array such as boolean operators and negation.
hashTags *array*
Filters to search for specific hash tags.
fromUser *string*
Filters for tweets sent by a specific user.
toUser *string*
Filters for tweets sent to a specific user.
place *string*
Filters tweets according to a specific place.
mentions *string*
Filters tweets according to whether a user was mentioned in it.
since_date *string*
Filters tweets by only returning those after a specified date.
attitude *string*
Filters tweets according to their attitude.
question *boolean*
Filters tweets according to if they are a question.
hasURL *boolean*
Filters tweets according to whether or not they contain a URL.
source *string*
Filters tweets according to how they were submitted.
customFilter *string*
A filter that allows for hand crafted queries.

Methods:

getQuery *string*
Purpose: To retrieve q.
Precondition: None
Postcondition: None
Parameters: None

setQuery *string*
Purpose: To allow users to set a custom search query.
Precondition: None
Postcondition: q should be populated with the query
Parameters: query a string that represents a custom query

1.4 ResultSet Class

This class describes a result set. The Query Class is used within the ResultSet Class.

1.4.1 ResultSet Class

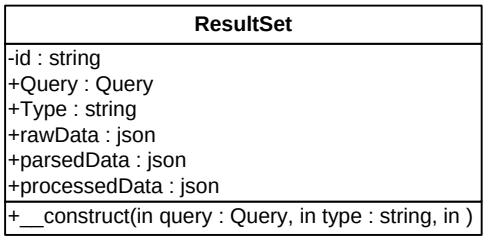


Figure 18 - ResultSet Class UML

Purpose:

This module holds all of the information regarding a specific query, algorithm, and the data returned.

Attributes:

id *string*
A unique identifier.
Query *Query*
The query object that generated this ResultSet.
Type *string*
The algorithm used to parse and process this data.
rawData *json*
The data retrieved from Twitter.
parsedData *json*
The data after being parsed.
processedData *json*
The data after being processed by the specified algorithm.

Methods:

__construct *none*
Purpose: The ResultSet constructor
Precondition: There must be a Query and a type.
Postcondition: There should be a ResultSet object.
Parameters: Query a query object, used to generate the data
type an algorithm type, which will be used for parsing and processing

1.5 AJAX Module

This module handles all of the requests between the client and the server. It has no submodules.

1.5.1 AJAX Module

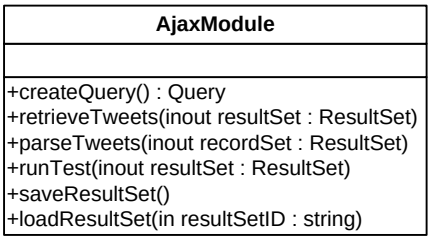


Figure 19 - AJAX Module UML

Purpose:

This module serves as a gateway so that the pages can interact with the various server modules.

Attributes:

None.

Methods:

createQuery *Query*
Purpose: To create a query object based on the data from the Query page.
Precondition: The submitQuery function must be filled with data from the QueryPage.
Postcondition: A query will be created.
Parameters: None

retrieveTweets *none*
Purpose: To send the query object to the tweet retrieval module and fill the resultSet's raw data.
Precondition: There must be a ResultSet created.
Postcondition: The raw data attribute of resultSet should contain retrieved data.
Parameters: resultSet a ResultSet which will have its raw data populated

parseTweets *none*
Purpose: To send the ResultSet to be parsed and fill the resultSet's parsed data.
Precondition: There must be a ResultSet created.
Postcondition: The parsed data attribute of resultSet should contain the data parsed as necessary, based upon its algorithm type.
Parameters: resultSet a ResultSet which will have its parsed data populated

runTest *none*
Purpose: To run an algorithmic analysis on a ResultSet
Precondition: There must be a ResultSet created.
Postcondition: The processed data attribute of resultSet should contain the data processed as necessary, based upon its algorithm type.
Parameters: resultSet a ResultSet which will have its parsed data populated

saveResultSet *none*
Purpose: Saves the current ResultSet into the database.
Precondition: There must be a ResultSet in the database.
Postcondition: The ResultSet should be saved in the database.
Parameters: None

loadResultSet *none*
Purpose: To load a specific ResultSet from the database into the current session.
Precondition: There must be a selected ResultSet id.
Postcondition: The specified ResultSet should be loaded into the current session.
Parameters: resultSetID a string containing the unique identifier for a ResultSet

1.6 Tweet Retrieval Module

This module handles search requests that are to be sent to Twitter. It has no submodules.

1.6.1 Tweet Retrieval Module

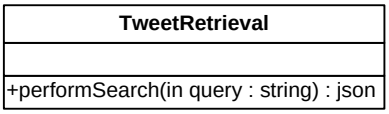


Figure 20 - Tweet Retrieval UML

Purpose:

This module retrieves data from Twitter.

Attributes:

None

Methods:

performSearch *json*

Purpose: To search Twitter for relevant tweets.

Precondition: The query must be in the form of a string.

Postcondition: The data should be retrieved in the json format to be handled.

Parameters: query a string, formatted according to the Twitter API

1.7 DataParser Module

This module handles parsing data retrieved from Twitter. It has no submodules.

1.7.1 DataParser Module

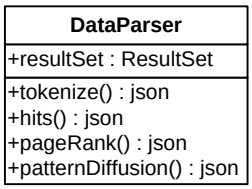


Figure 21 - DataParser UML

Purpose:

This module parses data retrieved from Twitter, depending on the algorithm being used to analyze it.

Attributes:

resultSet *ResultSet*

The resultSet contains information about the algorithm type so that the raw data can be parsed into a form usable for that algorithm.

Methods:

tokenize *json*

Purpose: To convert the tweets into a token based format, where only relevant data is contained.

Precondition: The resultSet must have type and rawData set.

Postcondition: The parsedData attribute should be set to the tokenized data.

Parameters: None

hits *json*

Purpose: To read the parsed data and do any more necessary parsing for the HITS algorithm.

Precondition: The resultSet must have the parsedData set.

Postcondition: The parsedData attribute should be set to the newly parsed data.

Parameters: None

Special Notes:

The HITS algorithm measures links between entities, originally designed for web pages. To parse the data, it will be necessary to measure the amount of in-degree links and out-degree links for any given tweet. Here, in-degree will be measured as the number of users that follow a tweet or user. Out-degree will be measured as the number of users that are following the tweet or the user that tweeted it. The parsing algorithm will be necessary to determine these numbers, and associate them with tweets.

pageRank json
Purpose: To read the parsed data and do any more necessary parsing for the pageRank algorithm.
Precondition: The resultSet must have the parsedData set.
Postcondition: The parsedData attribute should be set to the newly parsed data.
Parameters: None
Special Notes:
The PageRank algorithm is an evolution of the HITS algorithm, typically measuring the given probability that a user navigates to a specific web page. To parse the data, it will be necessary to determine the links between tweets and users. After these links are determined, it will be possible to build a directed graph so that the pageRank algorithm can be applied to the data.

patternDiffusion json
Purpose: To read the parsed data and do any more necessary parsing for use with diffusion geometries
Precondition: The resultSet must have the parsedData set.
Postcondition: The parsedData attribute should be set to the newly parsed data.
Parameters: None
Special Notes:
Pattern diffusion refers to the of categorizing data based on its diffusion geometries. This method finds diffusion distance by measuring distances between data based on similarities in multivariate characteristics. The algorithm will require the data be parsed to these characteristics. Examples of characteristics may include, but are not limited to: content, hashtags, users, and mood.

1.8 DataProcessor Module

This module handles processing data retrieved from Twitter. It has no submodules.

1.8.1 DataProcessor Module

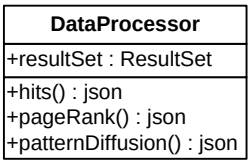


Figure 22 - DataProcessor UML

Purpose:

This module analyzes the Twitter data based on different algorithms.

Attributes:

hits *json*

Purpose: To read the parsed data and process it using the HITS algorithm.
Precondition: The resultSet must have the parsedData set.
Postcondition: The processedData attribute should be set to the processed data.
Parameters: None
Special Notes:
The HITS algorithm measures links between entities, originally designed for web pages. To process the data, it will be necessary to measure the amount of in-degree links and out-degree links for any given tweet. The HITS algorithm determines which tweets are hubs, or tweets that link to authority tweets. Authority tweets are determined by how many hubs link to the authority tweets. During this function, the tweets will be given hub and authority values which will rate their relative importance in each of those types.

pageRank *json*

Purpose: To read the parsed data and process it using the pageRank algorithm.
Precondition: The resultSet must have the parsedData set.
Postcondition: The processedData attribute should be set to the processed data.
Parameters: None
Special Notes:
The PageRank algorithm is an evolution of the HITS algorithm, typically measuring the given probability that a user navigates to a specific web page. To process the data, it will be necessary to determine the links between tweets and users. The algorithm will determine the importance of the tweets by replicating the path that a user may take if he was follows links from tweet to tweet. This will effectively find each tweet's relative importance by finding the likelihood that the user will be linked to that tweet.

patternDiffusion *json*

Purpose: To read the parsed data and process it for use with diffusion geometries
Precondition: The resultSet must have the parsedData set.
Postcondition: The parsedData attribute should be set to the newly parsed data.
Parameters: None
Special Notes:
Pattern diffusion refers to the of categorizing data based on its diffusion geometries. This method finds diffusion distance by measuring distances between data based on similarities in multivariate characteristics. The processing portion of the algorithm will take the categories, and determine diffusion distances for each of the tweets. These distances will be used to determine likeness between tweets.

2.0 Data Design

2.1 Data Architecture Diagram

This section details the structure of the data gathered.

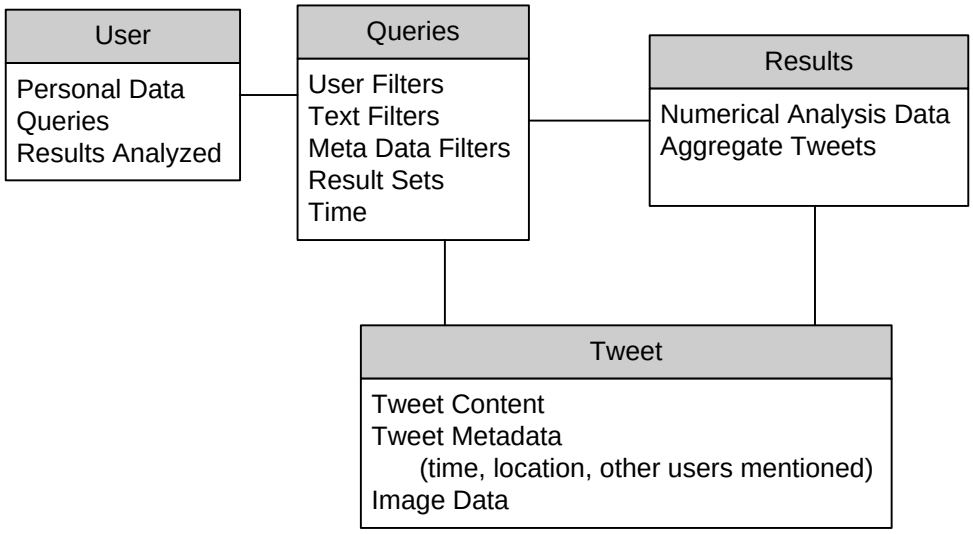


Figure 23 - Data Design

Specific user information will be recorded for demographics and login purposes. All text based data will be stored as nvarchar(max), while all numerical data will be stored as floats. Queries will be recorded, including the filters that the users chose and the result sets that apply to relevant queries. "User Filters" describes filters about the Twitter users' demographics. An example user filter may be age. Text filters are filters about specific content, such as words or phrases. A text filter may be searching for the word "Occupy". Meta data filters are set to only retrieve results that satisfy certain external criteria such as date and location. A meta data filter may be "Location: New York, NY. or Date: November 5, 2011" The results will contain the numerical data that is a result of the algorithm, as well as the tweets that make up a result set. A tweet will contain its content, as well as metadata such as time, location, other users, retweet status,

and hashtags. Additionally, any image data included in the tweet will be recorded. This image data will only be analyzed superficially to see if the same file is being retweeted. Future development may be able to make use of tools to track changes to the image or similarities, but that is outside the scope of the current project.

Chapter 6

Results and Analysis

Dataset

A typical data set consisted of 1000 Twitter messages. In order to gather appropriate attributes to describe the dataset, the method was as follows:

1. Retrieve the 1000 latest tweets from Twitter.
2. Iterate through each tweet, removing punctuation, and counting the frequency of each word that appears.
3. Eliminate all "stop words" -- common words in the English language.
4. Condense monetary values down into one attribute, "MONEY"
5. Determine the top 100 key words that appeared.
6. Iterate through each tweet, creating a vector of 100 attributes, where each attribute represents the frequency of a specific keyword appearing in the text. If a word appears with a hashtag in a tweet, it is given a frequency of three higher.

PCA Method

The PCA method is as follows:

1. Use the vector of attributes created by the above process to create a new matrix:

$A = X' \cdot X$

2. Determine the eigenvectors of A
3. The principle components are in ascending order, so the last two to three columns of the matrix are the ones that will be used to graph
4. K-means clustering based on the transpose of the original data set was used to determine how the tweets are categorized

Diffusion Geometry

The diffusion geometries method is as follows:

1. Determine the L2 distance
2. Determine the Gaussian kernel, K, as follows

$$K(x,y) = e^{\left(-\frac{\|x-y\|^2}{\sigma}\right)}, where\ \sigma\ was\ given\ at\ 10.0$$

3. Normalize K

$$K^{\alpha}(x,y) = \frac{K(x,y)}{p^{\alpha}(x)p^{\alpha}(y)}\ where\ p(x) = \sum K(x,y)P(y)dy$$

4. Apply weighted graph Laplacian

$$A(x,y) = \frac{K^{\alpha}(x,y)}{d^{\alpha}}\ where\ d^{\alpha} = \sum K^{\alpha}(x,y)P(y)dy$$

5. Compute SVD of A

In order to categorize users' tweets, each tweet message is represented as an attribute vector, which is made by the frequencies of certain keywords in the tweets' content. The top one hundred most frequently appearing meaningful words are picked as the keywords that each tweet would be represented by. It is impossible to display the clusters of the tweet vectors created by k-means because of the high dimensionality of the data. The various keywords may be related, so the intrinsic dimension of the data set is much smaller than one hundred.

A typical data set was created by searching for the latest 1000 tweets returned by a search for "democrat republican." The first three attributes are the key values of "republican", "democrat", and "rt"; these keys have the highest frequency across the entire data set. These top three attributes cannot be said to define the clusters as "#republican" and "republican" may appear frequently in the same tweet. Additionally, other words are likely correlated to those keywords. Figure 24 describes the tweets when graphed according to these three attributes.

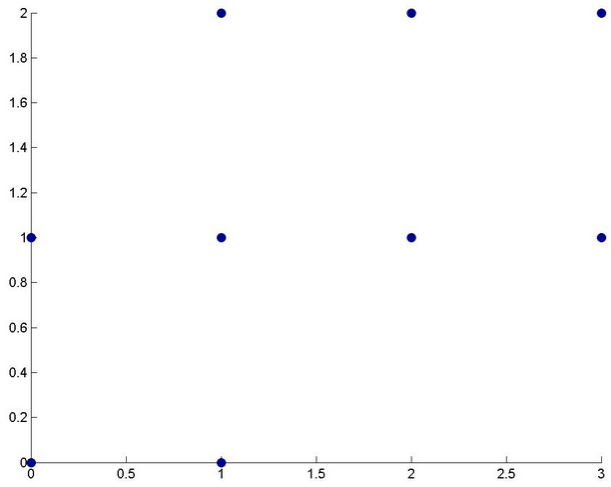


Figure 24 - First Three Attributes of Original Data

As the above figure illustrates, clusters cannot be identified; the data is very evenly spaced out. Similar results are found for other attributes chosen at random. Thus, the two methods described above, PCA and Diffusion Geometries, are used to reduce the dimensionality of the data.

The first of these methods, the PCA method, reduces the dimensionality of the data set down to three attributes. It is possible to reduce the dimensionality by more or less, but three was chosen for this example due to the ability to graph the data according to the new coordinate system. Figure 25 shows the same data set with the dimensions reduced according to the PCA method.

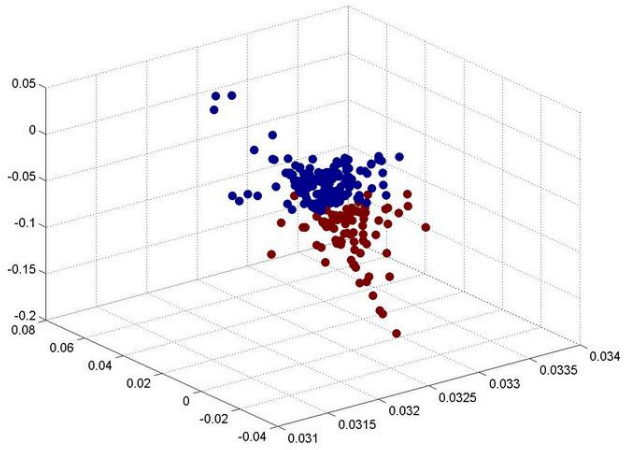


Figure 25 - PCA Method

It is clear which categories each tweet belongs to. In this case, there were two topics that were very popular, described by the two following tweets: " RT @OurPolls: The Internet is NOT a left or right, Democrat v. Republican issue. It's a basic human right issue and it is our future. #UnseatLamar", and " RT @washingtonpost: Is your dog a Republican or a Democrat? http://t.co/xNgNKuye". A similar tweet, such as: " 1 - Funny article. My dog is a GOP enthusiast, for sure. RT @washingtonpost: Is your dog a Republican or a Democrat? http://t.co/rpzTHmIA" " is much more similar to the latter, so it gets placed into to the same category.

A similar result can be seen using the diffusion map method. Figure 26 shows the result of the diffusion map method when clustered using k-means, based on the distances of the original tweets. Like the PCA method, two categories are easily filtered out. In this case, the red values correspond to the tweet containing the hashtag "UnseatLamar". In the data set, that category was very strongly defined due to a large appearance of anti-Lamar tweets; many of those tweets contained the specific phrase mentioned above.

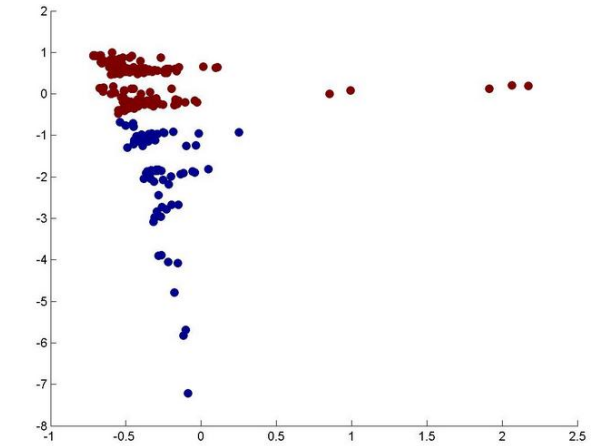


Figure 26 - Diffusion Map, Clustered by Original Data

If a k-means cluster is done on the data after the dimensionality has been reduced, a similar result is obtained.

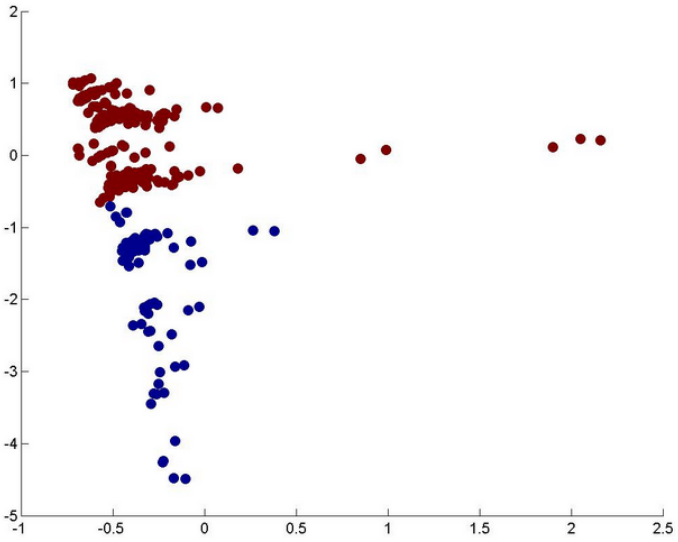


Figure 27 - Diffusion Map, Clustered by Diffusion Map Data

The tweets that changed color correspond to the tweets that were somewhere between the two. These are mostly tweets that have context pertaining to both topics. In this case, the twitter data

is a non-linear data set. PCA method works for this case due to the fact that the categories were so strongly identified by these two major topics.

These data mining techniques can be improved for text contained in social media. The first of many issues that appeared is noise in the data set. While basic data cleaning such as eliminating stop words, condensing all money amounts into one attribute, and weighting hash tags heavier was done, it may not be enough to clarify the data set. Instead, it may be appropriate to remove the keywords that were searched for, as well as words closely related. Additionally, it may be better to condense specific tenses or forms of a word down to one word. Also, typos and shortened words should be taken into account.

The algorithms could very easily be adjusted for this dataset. A clear example of this would be to modify the pattern diffusion algorithm to take multiple steps in the random walk instead of just the one that the algorithm was implemented with. Multiple iterations can help clarify the resemblances between the data.

The third major way that this method could be improved is through acquiring more data. The pattern diffusion method is much better as the dataset grows. With only 1000 tweets, the distances between the tweets may still be rather large, so there is a chance that tweets are categorized together that perhaps should not be. If nothing else, a larger dataset helps to confirm the existing results and increase the reliability that the method is correct.

The fact that the results for clustering are very similar for both the original data and the pattern diffusion data indicates that the method is proving to be effective at reducing the dimension of the data. With the above changes, pattern diffusion may prove to be an effective technique in categorizing tweets and other textual social media.

Chapter 7

References

Bakshy, E., Mason, W. A., Hofman, J. M., & Watts, D. J. (2011). Everyone's an Influencer: Quantifying Influence on Twitter. *Fourth International Conference on Web Search and Data Mining* .

Cha, M., Haddadi, H., Benevenuto, F., & Gummadi, K. (2010). Measuring User Influence in Twitter: The Million Follower Fallacy. *Proceedings of the Fourth International AAAI Conference on Weblogs and Social Media* .

Golbeck, J., & Rothstein, M. (2008). Linking Social Networks on the Web with FOAF. *AAAI'08 Proceedings of the 23rd national conference on Artificial intelligence - Volume 2* .

Haveliwala, T. H. (2003). Topic-Sensitive PageRank: A Context-Sensitive Ranking Algorithm for Web Search. *IEEE Transactions on Knowledge and Data Engineering* , 784-796.

Honey, C., & Herring, S. (2009, Januare 4-8). Beyond Microblogging: Conversation and Collaboration via Twitter. *System Sciences, 2009. HICSS '09. 42nd Hawaii International Conference on* , 1-10.

Huberman, B. A., Romero, D. M., & Wu, F. (2008, December 5). *Social Networks that Matter: Twitter Under the Microscope.* Retrieved August 28, 2011, from Social Science Research Network: http://papers.ssrn.com/sol3/papers.cfm?abstract_id=1313405

Java, A., Song, X., Finin, T., & Tseng, B. (2007). Why we twitter: understanding microbloggin usage and communities. *WebKDD/SNA-KDD '07* .

Kamvar, S., Haveliwala, T., & Golub, G. (2003). *Adaptive Methods for the Computation of PageRank.* Stanford.

Kleinberg, J. M., Kumar, R., Raghavan, P., Rajagopalan, S., & Tomkins, A. S. (1999). The Web as a graph: measurements, models, and methods. *Computing and Combinatorics* , 1-17.

Krishnamurthy, B., Gill, P., & Arlitt, M. (2008). A Few Chirps About Twitter. *WOSN '08 Proceedings of the first workshop on Online social networks* , 19-24.

Maggioni, M. (2004). *Diffusion Geometry*. Retrieved August 28, 2011, from Diffusion Geometry: http://www.math.duke.edu/~mauro/diffusiongeometries.html

Page, L., Brin, S., Motwani, R., & Winograd, T. (1198). *The PageRank Citation Ranking: Bringing Order to the Web.* Stanford Digital Library.

Appendix A

Glossary

Representation State Transfer (REST) -- REST is an architecture used for distributing information. These systems involve clients and servers, with a method to communicate within one another, often times with a client-side language, such as asynchronous javascript.

Trending Topic -- A word or phrase that has been posted by many users and has become one of the top posted and commented texts on twitter.

Academic Vita

Zachary Bush

4115 Cooper Road

Erie, PA 16510

zackdbush@gmail.com

Education: Bachelor of Science in Computer Science, Graduation Spring 2012

Honors in Computer Science

Thesis Title: Tweet Categorization with Pattern Diffusion

Experience: ExecutivePulse

Integrated graphing, mapping, and reporting functionality into business retention software. Assisted in the redesign of the web application.

Bullhead Research

Developing a web interface for database queries and reference sheet creation

Awards: Schreyers Honors College

Omicron Delta Kappa, National Honor Society

Lambda Sigma Sophomore Honor Society

Chancellor's Award

Deans List

Activities: Member of the PSU ACM ICPC Programming team

Study Abroad in Hong Kong (Chinese University of Hong Kong)

"graphMe"(*For Social Networking)*

Developed a facebook application to do graphical analysis of a personal social network