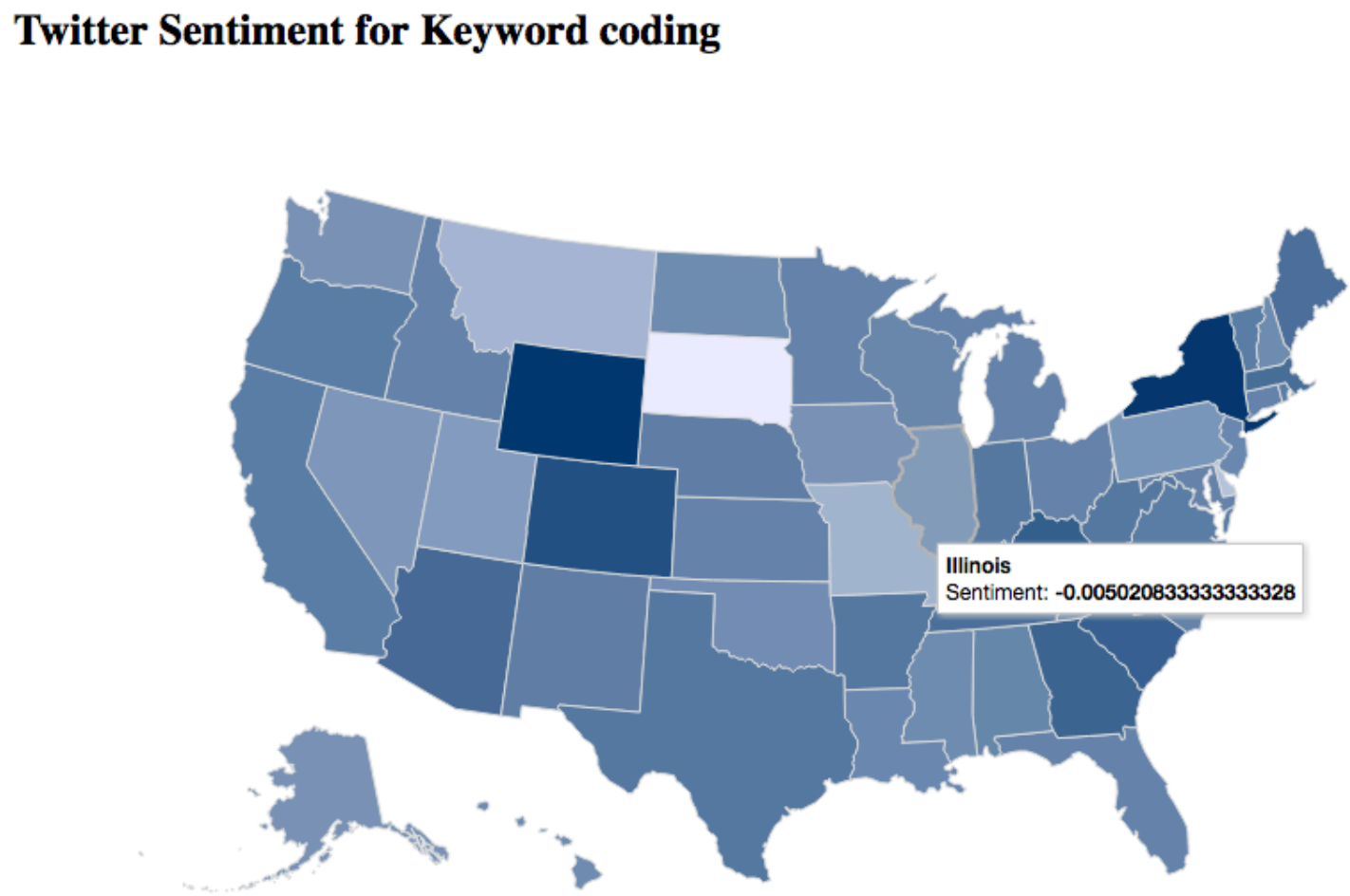
Twitter Mapping Lab

Introduction

Your application will allow the user to search Twitter for a particular word or phrase in tweets located in each of the 50 US states and then display on a map of the US the degree of positive or negative sentiment associated with that search term in each state. For example, if the search term is coding, the following map may be displayed.



This lab has several goals beyond the immediate concepts of decisions, loops, and unit tests in this unit:

- Exposure to data analytics. In this lab you will search a large data set (tweets on Twitter) and analyze that data to derive a new understanding (the sentiment of tweets containing a given keyword in each of the 50 US states).
- Experience using an API (Application Programming Interface) within the context of an unfamiliar software application. In this lab, you will use the Twitter4J API to access Twitter and write code within the partially implemented Twitter Mapping Lab application.
- Exposure to data visualization. In this lab you will visually represent the average sentiment in a map of the 50 US states.

Requirements

Background Reading

1. Read about APIs and why they are useful.
2. Read about sentiment analysis.

Twitter API Configuration

1. In the `TwitterMappingLab` folder is a `twitter4j.properties` file. This file will hold the credentials in order for you to access Twitter data.
2. If you don't already have a Twitter account, you need to make one. To make your own Twitter account, visit www.twitter.com and sign up. While you have to provide a valid email address as you will need to use it for verification, feel free to use a non-real username. You must add a mobile phone number to your Twitter profile in order to make a developer account later. You can also delete the account once the assignment is over, if desired.
3. Now that you have a Twitter account, you'll need to visit http://apps.twitter.com to set up the permissions to allow a non-Twitter app (your Java program) to access Twitter.
4. Select Create New App and fill in the first three fields of Application Details:
    a. Name: *name your project*
    b. Description: **Data Analytics and Visualization Lab**
    c. Website: *anything you want that starts with an http://*
    d. Callback URL: *leave blank*
5. Agree to the Developer Agreement.
6. Retrieve the four keys you will need for the `twitter4j.properties` file and edit that file by opneing it from within TextPad (change to browse all files):
    a. Click on "manage keys and access tokens"
    b. Click on "Create my access token"
    c. Update the `twitter4j.properties` file with your specific data.
    ```
    oauth.consumerKey=
    oauth.consumerSecret=
    oauth.accessToken=
    oauth.accessTokenSecret=
    ```
7. Let's see if you can tweet from your Java program! Type this code into a new `TestTweet` class:

```
import java.io.IOException;
import twitter4j.*;

class TestTweet
{
    public static void main ( String[] args ) throws TwitterException, IOException
    {
        Twitter twitter = new TwitterFactory().getInstance();
        twitter.updateStatus( "I just tweeted from my Java program! #APCSRocks Thanks @gcschmit!" );
    }
}
```

Tweet Class

1. Create a `Tweet` class to encapsulate a tweet.
2. The `Tweet` class will have the following instance variables: `user` (`String`), `text` (`String`), `location` (`twitter4j.GeoLocation`), `date` (`Date`), `sentiment` (`double`).

3. Write a constructor that takes as parameters the user, text, location, and date associated with the tweet. In the constructor invoke the `calculateSentiment` method on this class to determine the sentiment based on the text of the tweet.
4. Write accessors for all instance variables.
5. Write the `calculateSentiment` method which takes no parameters and returns no value. This method should calculate the sentiment associated with the text of the tweet based on the text of the tweet and with the assistance of the `SentimentDictionary` class that is provided as part of the Twitter Mapping Lab project. Calculate the sentiment of the tweet as the average of the sentiments for all the words in the tweet.
    a. To obtain a reference to a `SentimentDictionary` object, invoke the `getSingleton` static method on the `SentimentDictionary` class. See the documentation for more information.
    b. Use a `Scanner` object to iterate through each word in the text of the tweet.
    c. Use the `getSentiment` method to obtain the sentiment associated with each word. See the documentation for more information.
6. Write the `toString` method which returns a description of all the instance variables of this object.
7. Create a JUnit test class.
    a. Write a test method that tests the constructor and all accessors.
    b. Write a test method that tests the `calculateSentiment` method.

State Class

1. Create a `State` class to encapsulate the data associated with a state.
2. The `State` class will have the following instance variables: `abbreviation` (`String`), `center` (`twitter4j.GeoLocation`), `area` (`double`), `sentiment` (`double`).
3. Write a constructor that takes as parameters the abbreviation, center, and area associated with the state.
4. Write accessors for all instance variables.
5. Write the mutator method `setSentiment`.
6. Write the `getRadius` method which returns the radius for this state in units of miles, modeling the state as a circle.
7. Write the `toString` method which returns a description of all the instance variables of this object.
8. Create a JUnit test class.
    a. Write a test method that tests the constructor and all accessors.
    b. Write a test method that tests the mutator and accessor for the sentiment.
    c. Write a test method that tests the getRadius method.

TwitterMapper Class

1. The `TwitterMapper` class is partially implemented. Review the `loadStateInformation` and `mapSentimentForAllStates` methods to see what they do (they use concepts that we have not yet covered in class).
2. Implement the `findTweetsForState` method as described in the documentation.
    a. You will need to reference the Twitter4J API documentation and the general Twitter API documentation.
    b. The `getTweets` method of the `QueryResult` class returns a list of `Status` objects. To iterate through this list, use the following code, which we will explore later this semester:
    `for (Status status : result.getTweets())`
    This code will repeatedly assign a tweet to the local variable `status`.
3. Update the `main` method to perform the data analysis and visualization for the desired keyword.

Extensions:

- Improve the quality of a sentiment associated with a tweet (use emoticons?).
- Specify different operators in the Twitter query to look for different information.
- Analyze something other than the sentiment associated with a tweet.
- Visualize different information about a query.
- Add more awesome.

Submission:

- Ensure the following files have been committed to GitHub in addition to your code:
    - JUnit test code
    - HTML documentation generated by JavaDoc
    - Generated HTML files for more than one query
- Submit a link to your GitHub repository with this assignment.

Credit:

- This lab and parts of this document are based on Ria Galanos' Twitter Project.
- This lab is based on Stanford's Nifty Lab's Twitter Trends project by Aditi Muralidharan, John DeNero, and Hamilton Nguyen. The sentiment file included is from the Twitter Trends project.
- The visualization is done using the DataMaps Javascript framework. Specifically, the HTML file is based on the Choropleth with auto-calculated color example.
- The data for the geographic center and area of each state was obtained from Wikipedia.