Northwestern

GeoTweet!

Jeff Brown | Eugenio Gallastegui |
Saira Jahangir | Kamil Borowik | Alan Garbarino

GeoTweet II

- GeoTweet I - (project 2)
- Enhancements
- Backend/Data Management
- Demographic Charting
- Sentiment Analysis
- Machine Learning

GeoTweet I – Project 2

Interactive map with overlay of Twitter trending locations

GeoJSON State boundaries with demographic color scaling based on political affiliation and popups providing US Census statistics

Dynamic table update showing twitter trends by location selected

Responsive scatter plot charts showing demographic data against political affiliation at State level for trending locations

Flask application, SQLite database, Heroku platform

- HTML/CSS, Javascript Leaflet, GeoJSON, D3, Plotly, Heroku, SQLite

GeoTweet Enhancements

Enhanced User Experience

New trend dimension – select a trend (from table) to see all trend locations on map

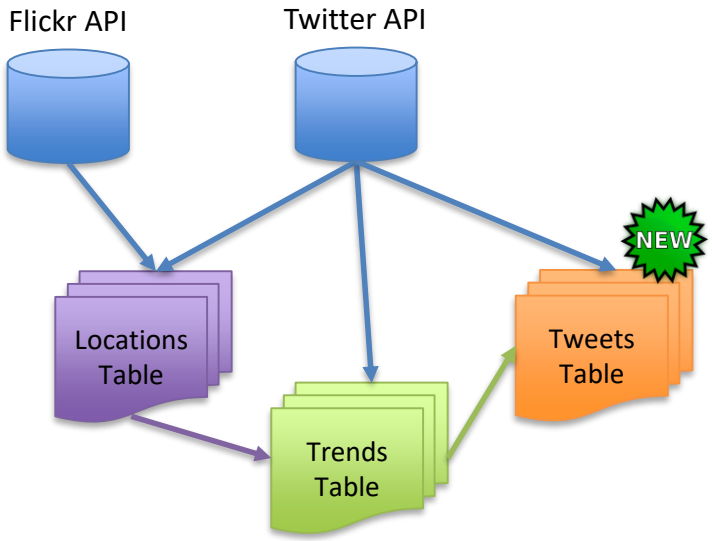
Trend persistence – Up to 3 trends selections built as independent overlays, recycle first overlay when 4th trend selected.

Responsive feedback – D3 chained transition to indicate when a trend is clicked.

- HTML/CSS-Javascript-D3

Backend/Data Management

Enhanced to Support Time-Series and Tweet/User-level Analysis



➢ Flask-Postgres-Heroku

- **Database/Backend Enhancements**
  - **Tweets table added**: Tweet/User data
  - **Multiple snapshots over time** of Trends table entries
  - **Retention over time** of Locations and Tweets data
- **21 Flask Routes**
  - **Queries** of Locations, Trends, and Tweets tables
  - **Rendering** of Main and Demographics pages
  - **Update** of Sentiment Analysis, Table Data, Table Status
- **Support for SQLite and Postgres** for development
- **Deployed on Heroku**
  - Dyno – geotweetapp
  - Postgres Database – Up to 10M records
  - Scheduler – Automated updates daily (locations) and hourly (trends and tweets)

Demographic Charts

Social Context for tweets

Retrieve demographic data from the US census.

Concentrated analysis on employment, education attainment level and political party affiliation (Democratic/Republican) by State.

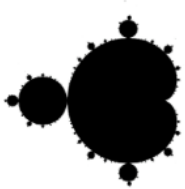
Modified design to move State overview charts to new Demographics page.

- Javascript-Plotly-D3

Sentiment Analysis

This feature takes the user's input 

that triggers the function `def DownloadData(self):` that works within the class `class SentimentAnalysis:` .



Said function uses TextBlob , a user-friendly API for Text and Natural Language Processing.

Anatomy of `class SentimentAnalysis:`

1. Takes user's input to trigger the process 

2. Access the Twitter Data using Twitter API Keys and Tweepy
3. Retrieve the hashtags that match the user's input.
4. Process the content that accompanies the hashtag using TextBlob
5. With the analysis results creates a plot using plotly
6. Sends the plot from the cloud to the app using an `<iframe>` embedded into the `index.html`

https://github.com/Euzkaro/Project2.io/blob/master/resources/Sentiment_Analysis.ipynb

Voilà!

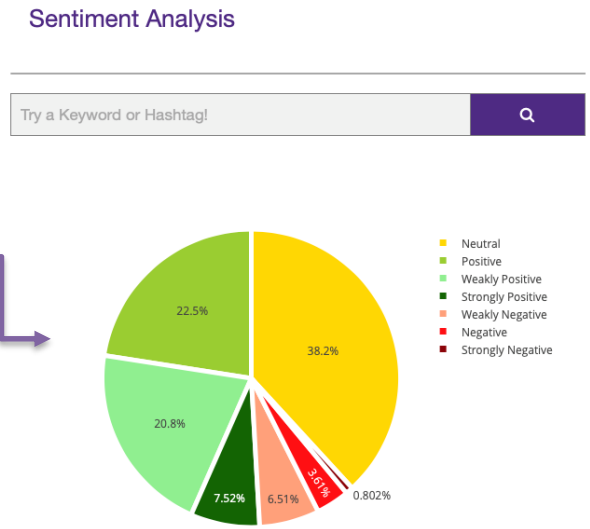
Py Flask Plotly / HTML

```
for tweet in self.tweets:
    #Append to temp so that we can store in csv later. Encode in UTF-8
    self.tweetText.append(self.cleanTweet(tweet.text).encode('utf-8'))
    analysis = TextBlob(tweet.text)
    #print(analysis.sentiment)  # print tweet's polarity
    polarity += analysis.sentiment.polarity  # adding up polarities to find the avera

    if (analysis.sentiment.polarity == 0):  # adding how people are reacting to find
        neutral += 1
    elif (analysis.sentiment.polarity > 0 and analysis.sentiment.polarity <= 0.3):
        wpositive += 1
    elif (analysis.sentiment.polarity > 0.3 and analysis.sentiment.polarity <= 0.6):
        positive += 1
    elif (analysis.sentiment.polarity > 0.6 and analysis.sentiment.polarity <= 1):
        spositive += 1
    elif (analysis.sentiment.polarity > -0.3 and analysis.sentiment.polarity <= 0):
        wnegative += 1
    elif (analysis.sentiment.polarity > -0.6 and analysis.sentiment.polarity <= -0.3)
        negative += 1
    elif (analysis.sentiment.polarity > -1 and analysis.sentiment.polarity <= -0.6):
        snegative += 1


# finding average of how people are reacting
positive = self.percentage(positive, NoOfTerms)
wpositive = self.percentage(wpositive, NoOfTerms)
spositive = self.percentage(spositive, NoOfTerms)
negative = self.percentage(negative, NoOfTerms)
wnegative = self.percentage(wnegative, NoOfTerms)
snegative = self.percentage(snegative, NoOfTerms)
neutral = self.percentage(neutral, NoOfTerms)
```

```
@app.route("/sentiment/<search_input_global>")
def sentiment_input():
    logger.info("Are we here?")
    if search_input_global != "":
        sa = SentimentAnalysis()
        sa.DownloadData(search_input_global)
```



Machine Learning

Predictive Analytics (Part I)

Predict if a user will be an influencer (> 100 retweets) based on volume in the following user categories

tweet_user_statuses_count: count - user tweets
tweet_user_favourites_count: count - user is a favorite of others
tweet_user_followers_count: count - user is followed by others
tweet_user_friends_count: count - user friends
tweet_user_listed_count: count - user added to other's interest list

Training Data Score: 0.7653103677816608 Testing Data Score: 0.7660052414825907

- Python(Sci-Kit Learn)-Logistic Regression (AIevalSQLite-Scale.ipynb)

Machine Learning

Predictive Analytics (Part II)

Predict using the Scikit-Learn library -> are there any trends in the data sets?

Retrieve tweets by Twitter Categories and perform text pre-processing to convert textual data to numeric data by counting words for each category

Finally, use machine learning algorithms to train and test our model.

Linear Regression Model : lr.score -> 0.676, Random Forest Classifer : lr.score -> 0.75

- Python(Sci-Kit Learn, NLK) Linear Regression-Random Forest

Questions