Twitter Based GeoSentiment Analysis In the Sacramento Region

Produced for ARC: Los Rios Geo 375

M. Hernandez, PhD

Motivations

Why do we care

Empowering organizations with the knowledge of not just what people are talking about, but where the conversation is and how do people feel about it is the a real challenge.

What can be done

Leverage various open source pieces of information to reach inferential conclusions on these unanswered questions.

Who is influencing

Identify who is potentially influencing these trends and where are their areas of influence.

Analysis Pipeline

Tweet acquisition is done by geospatially associating tweets with twitters geofencing options that they provide in their api

As tweets are being pulled from the database a sentiment analysis program is tested against the indexed tweets

Now that the tweets are related to both a geographic region and a determined to being positive or negative information is rendered by arcpy

Technologies Involved

The main technologies involved in this project are ArcGIS and Python with the incorporation of essential modules

- Python
- ArcGIS

Technologies Involved Cntd.

- Python
  - Programmatic language to engage with ArcGIS, Twitter, and Sentiment analysis
- ArcGIS
  - Platform used for presenting geospatially dependent information

Technologies Involved Cntd.

Specific Python Packages

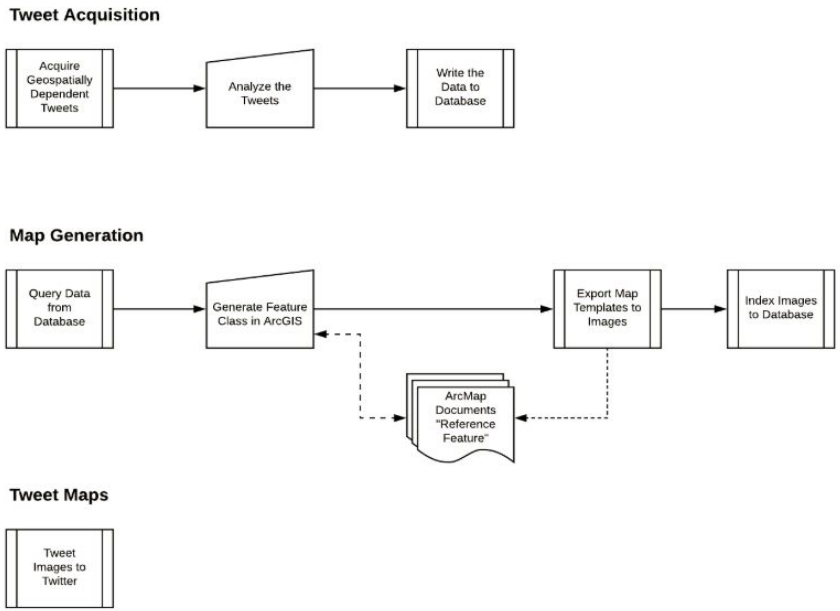
- arcpy: ArcGIS communication
- tweepy: connector to Twitter
- textblob: sentiment analysis module
- nltk: preprocessing of textual information
- sqlalchemy: used to manage and create datamodel
- sqlite3: used to insert and query data from sqlite database

Overview of the Project Workflow

Three main parts to the project workflow

- Acquisition of spatially related Tweets; done independent of the following steps
- Map Generation of Tweet "sentiment"
- Tweet sentiment encoded maps to Twitter

Diagram of Workflow



Tweet Acquisition

Python module: tweepy

- Module used to interact with twitter

Grid out a specified geographic region

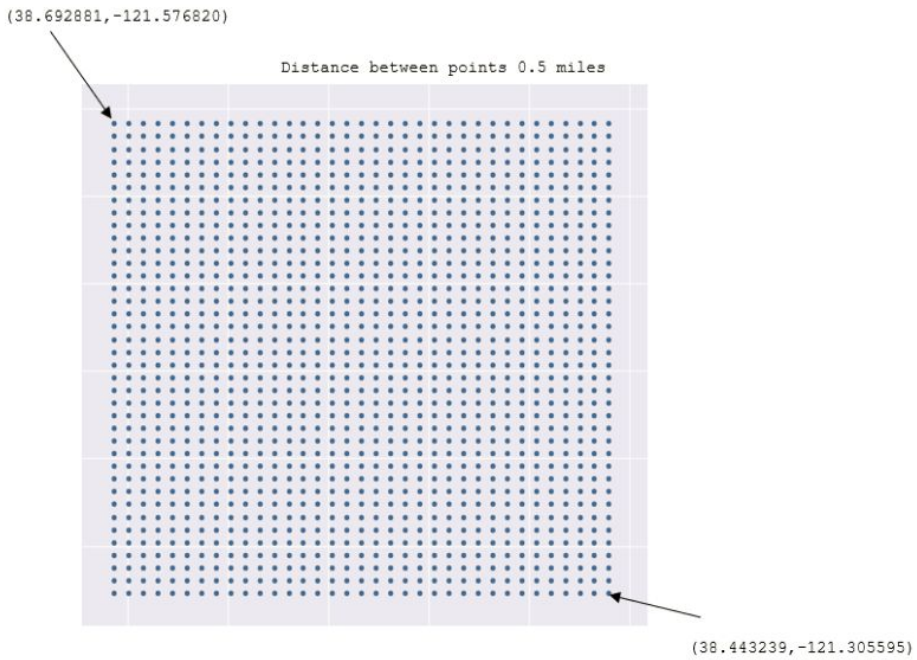
- This project chose the northwest corner of analysis (38.692881,-121.576820) and the southeast corner to be (38.443239,-121.305595) with a 0.5 mile radius

Tweet Acquisition

Grid out a region

- This is done to indirectly measure tweets that are found within a geographic region
- You can pay for a service OR you can infer by passing a geographic point with a tolerance radius to the twitter api
- This allows you to obtain tweets related to a specific geographic region

Tweet Acquisition: Tweet Grid



Tweet Acquisition: Sentiment Analysis

Python module: textblob

We utilize textblob as an example of a way to estimate sentiment.

Note: The implementation of this module is meant as an example. In real world situations there are many additional steps in order to generate an accurate and reliable sentiment estimator

Tweet Acquisition

Data Model | SQLAlchemy

DB creation managed by SQLAlchemy

- tweet table holds the tweets that are streamed and analyzed
  - id : primary key
  - text : text from tweet that is analyzed
  - sentiment : positive | negative | neutral
  - latitude : latitude of the tweet
  - longitude : longitude of the tweet
- image_tweets table that holds the images of the maps that are created
  - tweet_image_id : primary key
  - filename : filepath to file
  - params : field that you can use to populate for criteria
  - processed : flag for if the image has been uploaded
  - twitter_id : id of the tweet associated with the upload

```python
import os
import sys
from sqlalchemy import Column, ForeignKey, Integer, String, Text, Float, BLOB
from sqlalchemy.ext.declarative import declarative_base
from sqlalchemy.orm import relationship
from sqlalchemy import create_engine
import datetime

Base = declarative_base()


class Tweet(Base):
    __tablename__ = 'tweet'
    id = Column(Integer, primary_key=True, index=True)
    text = Column(Text, nullable=False)
    sentiment = Column(Text, nullable=True)
    latitude = Column(Float)
    longitude = Column(Float)


class TweetImage(Base):
    __tablename__ = 'image_tweets'
    tweet_image_id = Column(Integer, primary_key=True, index=True)
    filename = Column(Text)
    params = Column(Text)
    created_date = Column(Text, default=datetime.datetime.now().isoformat())
    processed = Column(Text, server_default='0')
    twitter_id = Column(Text, nullable=True, server_default='')


engine = create_engine('sqlite:///sqlite.db')
Base.metadata.create_all(engine)
```

Tweet Acquisition

Sentiment & Tweets

We pass a geographic point to the function to obtain the tweets the within a 0.5 mile radius

For each tweet that we retrieve for that given specified region we pass the text through our sentiment analyzer

* special thanks to https://www.geeksforgeeks.org/twitter-sentiment-analysis-using-python/ for making my life easier by detailing a great example

```
def get_tweets(self, query, count=100, point=None):
    '''
    Main function to fetch tweets and parse them.
    '''
    # empty list to store parsed tweets
    tweets = []

    try:
        # call twitter api to fetch tweets
        if point is None:
            fetched_tweets = self.api.search(q=query, count=count, geocode='{geo},{radius}'.format(
                geo=settings.geo_code, radius=settings.radius))
        else:
            fetched_tweets = self.api.search(q=query, count=count, geocode='{lat},{lon},0.5mi'.format(
                lat=point[0], lon=point[1]))
        # parsing tweets one by one
        for tweet in fetched_tweets:\
            # empty dictionary to store required params of a tweet
            parsed_tweet = {}

            # saving text of tweet
            parsed_tweet['text'] = self.clean_tweet(tweet.text)
            parsed_tweet['id'] = tweet.id
            # saving sentiment of tweet
            parsed_tweet['sentiment'] = self.get_tweet_sentiment(tweet.text)

            # appending parsed tweet to tweets list
            if tweet.retweet_count > 0:
                # if tweet has retweets, ensure that it is appended only once
                if parsed_tweet not in tweets:
                    tweets.append(parsed_tweet)
            else:
                tweets.append(parsed_tweet)

        # return parsed tweets
        return tweets
        # return fetched_tweets

    except tweepy.TweepError as e:
        # print error (if any)
        print("Error : " + str(e))
```

Twitter Acquisition

Streaming Tweets

We leverage the previously defined tools to periodically check for new tweets at each predefined geographic location.

*Due to twitter restrictions on their api we need to make sure that we put a hard wait in before cycling through the geographic extent

```python
import sentiment
import settings
from models import *
from sqlalchemy.orm import sessionmaker
import time


def main():
    api = sentiment.TwitterClient()
    full_set = []
    engine = create_engine('sqlite:///sqlite.db')
    Base.metadata.bind = engine
    DBSession = sessionmaker(bind=engine)
    for i in range(len(settings.gps_array)):
        try:
            session = DBSession()
            time.sleep(10)
            tweets = api.get_tweets(query=' ', point=settings.gps_array[i])
            clean_tweets = []
            if tweets is not None:
                for tweet in tweets:
                    tweet['latitude'] = settings.gps_array[i][0]
                    tweet['longitude'] = settings.gps_array[i][1]
                    try:
                        # print tweet
                        clean_tweets.append(tweet)
                    except:
                        pass
                full_set += clean_tweets
                if len(clean_tweets) > 0:
                    new_tweets = []
                    for tweet in clean_tweets:
                        if session.query(Tweet).filter_by(id=tweet['id']).first() is None:
                            new_tweets.append(Tweet(**tweet))
                        else:
                            pass
                    if len(new_tweets) > 0:
                        session.add_all(new_tweets)
                        session.commit()
        except:
            pass


if __name__ == '__main__':
    while True:
        main()
        time.sleep(900)
```

Map Tweets

Create sentiment feature

- creates new feature class “SentimentEstimate”
  - deletes feature class if it exists
  - adds fields to feature class
  - leverages proper spatial referencing within the map
  - specifies point feature
- Iterates over each of the fields and adds them to the feature based on the settings that we define in the fc_fields list of dictionary items
- We declare a min_sent and max_sent that we will use to capture extremum values

```
def make_map(start_date=None, end_date=None):
    if start_date is None:
        beg_date = settings.start_date
        final_date = settings.end_date
    else:
        beg_date = start_date
        final_date = end_date
    try:
        fc_workspace = os.path.join(os.getcwd(), 'out_fgdatabase.gdb')
        fc_name = "SentimentEstimate"
        outpath_table = os.path.join(fc_workspace, fc_name)

        fc_fields = [dict(in_table=None,
                          field_name='tweet_id',
                          field_type='TEXT',
                          field_length=100),
                     dict(in_table=None,
                          field_name='latitude',
                          field_type='FLOAT'),
                     dict(in_table=None,
                          field_name='longitude',
                          field_type='FLOAT'),
                     dict(in_table=None,
                          field_name='sentiment',
                          field_type='TEXT',
                          field_length=50),
                     dict(in_table=None,
                          field_name='SentimentValue',
                          field_type='FLOAT')
                    ]

        arcpy.env.workspace = fc_workspace

        if arcpy.Exists(outpath_table):
            arcpy.Delete_management(outpath_table)

        sr = arcpy.SpatialReference(4326)

        fc = arcpy.CreateFeatureclass_management(fc_workspace, fc_name, "POINT", spatial_reference=sr)

        for fc_field in fc_fields:
            fc_field['in_table'] = fc
            arcpy.AddField_management(**fc_field)

        fields = [val['field_name'] for val in fc_fields]
        fields.append("SHAPE@XY")
        conn = sqlite3.connect('sqlite.db')
        min_sent = 0
        max_sent = 0
```

Map Tweets

Insert the tweets into the newly created feature class

- Queries database and summarizes sentiment value based on geospatial reference and a date range
  - the default date range for automation is 1 day back
- identify if the sentiment value is positive, negative, or neutral; (neutral here is none)
- Check if the current inserted value is a new maximum or new minimum and if so re-assign the value

```
with arcpy.da.InsertCursor(outpath_table, fields) as irows:
    query = '''select 'id' as id, latitude, longitude, sum(case when sentiment = 'positive'
    then 1 else case when sentiment = 'negative' then -1 else 0 end end) as sent_value
    from tweet
    where created_date > '{beginning_date}'
    and created_date < '{end_date}'
         group by 1,2,3
    '''.format(beginning_date=beg_date, end_date=final_date)
    tweets = conn.execute(query).fetchall()
    for tweet in tweets:
        if tweet[3] > 0:
            sent = 'positive'
        elif tweet[3] < 0:
            sent = 'negative'
        else:
            sent = 'none'
        irows.insertRow(
            (str(tweet[0]),
             round(tweet[1], 6),
             round(tweet[2], 6),
             str(sent),
             tweet[3],
             (round(tweet[2], 6), round(tweet[1], 6)))
        )
        if tweet[3] < min_sent:
            min_sent = tweet[3]
        if tweet[3] > max_sent:
            max_sent = tweet[3]
```

Map Tweets

Full Map with positive and negative tweets

- Uses arcmap's tweet_template.mxd as map document
- Updates symbology based on sentiment value
- Updates the title for the date range of values that are being used
- Exports the map document to a jpg and stores to image directory
- Inserts a record in the database for the newly created image and flagged as 0 for not being uploaded

```
# full map tweet
mxd = MapDocument(os.path.join(settings.gis_template_directory, 'tweet_template.mxd'))
for lyr in arcpy.mapping.ListLayers(mxd):
    try:
        if lyr.symbologyType == "GRADUATED_COLORS":
            lyr.symbology.valueField = "SentimentValue"
            lyr.symbology.classBreakValues = [min_sent, 0, 0.000001, max_sent]
            lyr.symbology.classBreakLabels = ["{min} to 0 : Negative".format(min=round(min_sent, 0)), "0 : Neutral",
                                              "1 to {max} : Positive".format(max=round(max_sent, 0))]
    except:
        pass

titleItem = arcpy.mapping.ListLayoutElements(mxd, "TEXT_ELEMENT")[0]
titleItem.text = "Sacramento Tweet Sentiment\n {start_date} - {end_date}".format(start_date=beg_date,
                                                                                 end_date=final_date)
map_image_name = os.path.join('images', str(round(time.time() * 1000)) + '.jpg')
arcpy.mapping.ExportToJPEG(mxd, map_image_name)
sql_insert_statement = '''INSERT INTO image_tweets
    (filename, params, created_date)
    VALUES
    ('{name}', '{params}', '{date}')'''.format(name=map_image_name, params=str({"start_date": str(settings.start_date)}).replace("'
                                               date=datetime.datetime.now().isoformat())
conn.execute(sql_insert_statement)
conn.commit()
```

Map Tweets

Positive | Negative Maps

- Each arcmap document has a built in definition layer query for the feature class SentimentValue
  - negative sentiment for negative
  - positive sentiment for positive
- Title is updated with the proper date range
- Symbology is updated based upon the extremum values that were being tracked during the inserting of the records into the database
  - larger radius corresponds with a more positive sentiment value
- Map is exported as a jpg to the images directory
- Inserts a record in the database for the newly created image and flagged as 0 for not being uploaded

```
# negative tweets
mxd = MapDocument(os.path.join(settings.gis_template_directory, 'negative_template.mxd'))
for lyr in arcpy.mapping.ListLayers(mxd):
    try:
        if lyr.symbologyType == "GRADUATED_SYMBOLS":
            lyr.symbology.valueField = "SentimentValue"
            lyr.symbology.numClasses = 3
    except:
        pass

titleItem = arcpy.mapping.ListLayoutElements(mxd, "TEXT_ELEMENT")[0]

titleItem.text = "Sacramento Negative Tweet Sentiment\n {start_date} - {end_date}".format(start_date=beg_date,
                                                                                         end_date=final_date)

map_image_name = os.path.join('images', str(round(time.time() * 1000)) + '.jpg')
arcpy.mapping.ExportToJPEG(mxd, map_image_name)
sql_insert_statement = '''INSERT INTO image_tweets
           (filename, params, created_date)
           VALUES
           ('{name}', '{params}', '{date}')'''.format(name=map_image_name, params=str(
    {"start_date": str(settings.start_date)}).replace("'", '"'),
                                                      date=datetime.datetime.now().isoformat())
conn.execute(sql_insert_statement)
conn.commit()

# positive tweets
mxd = MapDocument(os.path.join(settings.gis_template_directory, 'positive_template.mxd'))
for lyr in arcpy.mapping.ListLayers(mxd):
    try:
        if lyr.symbologyType == "GRADUATED_SYMBOLS":
            lyr.symbology.valueField = "SentimentValue"
            lyr.symbology.numClasses = 3
    except:
        pass
titleItem = arcpy.mapping.ListLayoutElements(mxd, "TEXT_ELEMENT")[0]
titleItem.text = "Sacramento Positive Tweet Sentiment\n {start_date} - {end_date}".format(start_date=beg_date,
                                                                                         end_date=final_date)
map_image_name = os.path.join('images', str(round(time.time() * 1000)) + '.jpg')
arcpy.mapping.ExportToJPEG(mxd, map_image_name)
sql_insert_statement = '''INSERT INTO image_tweets
           (filename, params, created_date)
           VALUES
           ('{name}', '{params}', '{date}')'''.format(name=map_image_name, params=str(
    {"start_date": str(settings.start_date)}).replace("'", '"'),
                                                      date=datetime.datetime.now().isoformat())
conn.execute(sql_insert_statement)
conn.commit()
```

Tweet Map

Upload Map Images

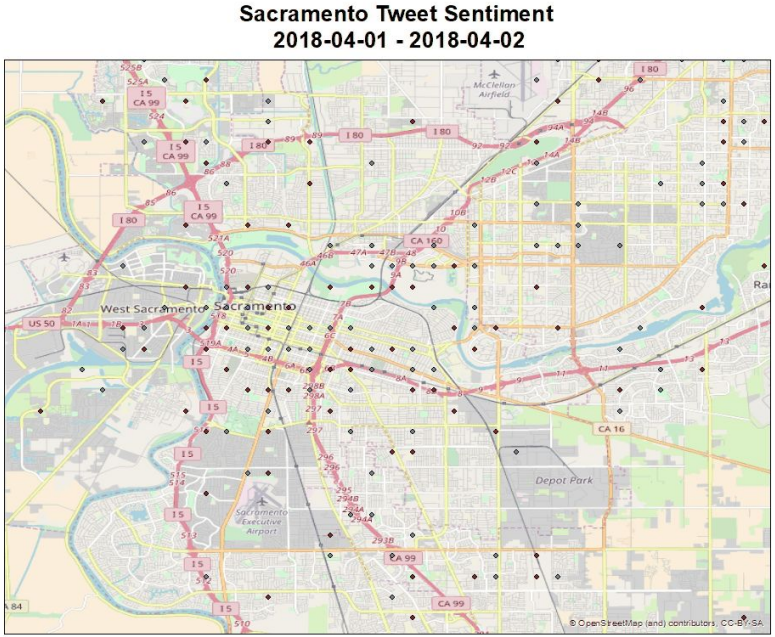
- Stand alone script that queries the local database for any image that was created and has not been uploaded
- If there are images that have not been uploaded then each file is uploaded to twitter
- After the file has been uploaded to twitter, the associated tweet_id is used to update the record in the database and the flag for the image is changed to processed

```
import os
import settings
from sentiment import TwitterClient
import sqlite3
import time
import tweepy


def upload_tweet():
    conn = sqlite3.connect(settings.database_name)
    images = conn.execute('''
        select
            tweet_image_id,
            filename
        from
            image_tweets
        where
            processed = 0 ''').fetchall()
    api = TwitterClient()
    if len(images) > 0:
        for image in images:
            time.sleep(3)
            id = image[0]
            fname = os.path.join(os.getcwd(), image[1])
            resp = api.api.update_with_media(filename=os.path.join(os.getcwd(), fname), status=settings.twitter_status_update)
            if resp.id is not None:
                conn.execute('''UPDATE image_tweets
                                SET processed = 1,
                                twitter_id = {tweet_id}
                                WHERE tweet_image_id = {id}
                            '''.format(tweet_id=str(resp.id), id=id))
                conn.commit()


if __name__ == '__main__':
    upload_tweet()
```

Production

Stream tweets

- Run synch_db.py
  - This is a standalone process that needs to be started once
- Schedule arcpy_create_feature_class.py
  - This is should be schedule once a day
- Run upload_tweet.py
  - This can be done throughout the day since this checks for new records to upload to twitter



Acknowledgements

Thank you to the professors and lecturers at ARC Los Rios GIS Program.

Special thanks to Nathan Jennings for covering arcpy techniques and implementation methodologies within this course, Professor Howard for providing feedback throughout the project, Levi Lebhart for sharing input on map design, Christopher Michel for entertaining my bad ideas, and my wife Anastasiya for her continuous support.

Appendix

All code can be found at

https://github.com/gis-delosrios/final_project

Home page of project can found at

https://sites.google.com/apps.losrios.edu/geog-375-final/home