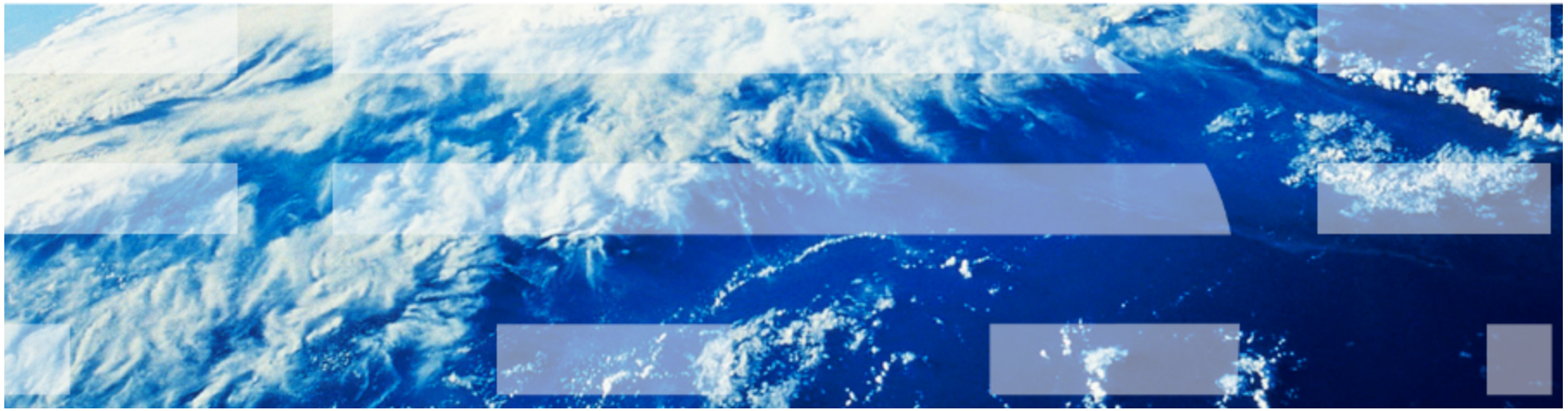
E6893 Big Data Analytics:

Tweets analysis and area safety prediction

Team Members: Heng Ji, Wanheng Li, Long Long

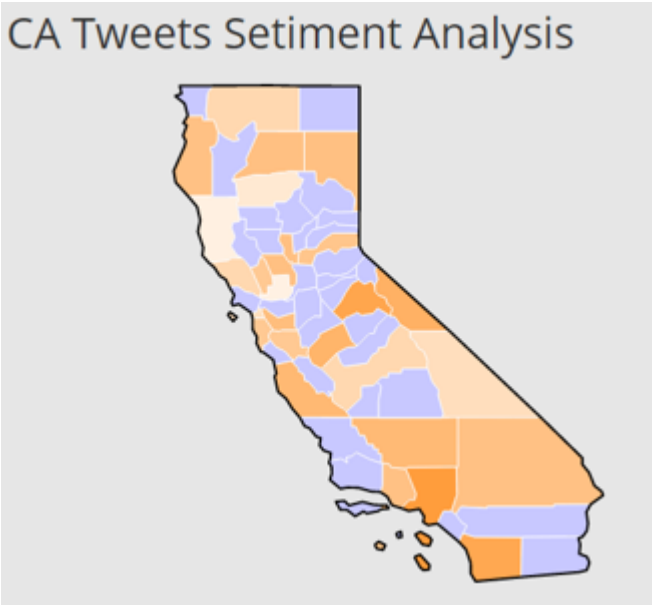


December 15, 2016

Overview

Tweets has become a very efficient method to reflect people's moods. We will extract features from tweets and classify each tweets into positive and negative groups.

Use collected data to reflect the order of a specific area. Find the correlation between tweets and local safety. Accordingly, predict local safety factor and recommend police force.



Technologies Used

Columbia University

Details | Settings | Keys and Access Tokens | Permissions

data mining

https://www.ee.columbia.edu/~cylin/course/bigdata/

Organization

Information about the organization or company associated with your application. This information is optional.

APIs

Programmatic access to read and write Twitter data

Learn about APIs

Apache Spark™

tweepy

v3.5.0

D3

Dataset Overview

We mainly have two separate sets of data:

- City Open Public Safety Data

  Collected by police departments of major cities including date, time, incident type, geo location…

NYC OpenData

LOS ANGELES OPEN DATA

Information, Insights, and Analysis from the City of Los Angeles

NYPD Complaint Data Historic OFFICIAL Public Safety

View Data | Download | API | Share | ...

This dataset includes all valid felony, misdemeanor, and violation crimes reported to the New York City Police Department (NYPD) from 2006 to the end of last year (2015). For additional details, please see the attached data dictionary in the 'About' section.

Updated
November 22, 2016

Data Provided by
Police Department (NYPD)

What's in this Dataset?

Rows	Columns
5.1M	24

Columns in this Dataset

Column Name	Description	Type
CMPLNT_NUM		Number #
CMPLNT_FR_DT		Date & Time
CMPLNT_FR_TM		Plain Text T
CMPLNT_TO_DT		Date & Time
CMPLNT_TO_TM		Plain Text T
RPT_DT		Date & Time

Table Preview

Explore Data

PREM_TYP_DESC	PARKS_NM	HADEVELOPT	X_COORD_CD	Y_COORD_CD	Latitude	Longitude	Lat_Lon
RESIDENCE - APT. HOUSE			995,500	149,215	40.576237762	-73.959504022	(40.576237762°, -73.9595
STREET			1,017,933	232,218	40.804004719	-73.878335461	(40.804004719°, -73.8783
STREET			1,012,778	255,259	40.867263716	-73.896857767	(40.867263716°, -73.8968
OTHER			1,054,851	181,458	40.664463748	-73.745517566	(40.664463748°, -73.7455
RESIDENCE - APT. HOUSE			1,023,475	207,175	40.735245416	-73.858463451	(40.735245416°, -73.8584
DEPARTMENT STORE			998,954	233,791	40.808374136	-73.946885823	(40.808374136°, -73.9468
STREET			1,011,365	192,916	40.696153306	-73.902218102	(40.696153306°, -73.9022
DEPARTMENT STORE			987,220	212,676	40.750430768	-73.989282176	(40.750430768°, -73.9892
RESIDENCE-HOUSE			1,045,862	187,367	40.680749693	-73.777864583	(40.680749693°, -73.7778

Dataset Overview

We mainly have two separate sets of data:

- Tweets with geo location

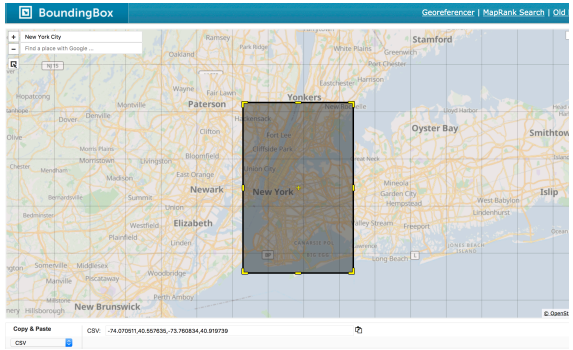
  Used twitter API and tweepy to collect tweets sent out from selected cities

locations

A comma-separated list of longitude,latitude pairs specifying a set of bounding boxes to filter Tweets by. Only geolocated Tweets falling within the requested bounding boxes will be included—unlike the Search API, the user's location field is not used to filter tweets.

Each bounding box should be specified as a pair of longitude and latitude pairs, with the southwest corner of the bounding box coming first. For example:

Parameter value	Tracks Tweets from...
-122.75,36.8,-121.75,37.8	San Francisco
-74,40,-73,41	New York City
-122.75,36.8,-121.75,37.8,-74,40,-73,41	**San Francisco** OR New York City



Challenges & Struggles

1. Feature method: Using Tf-idf, Bigram, Unigram or Combination.

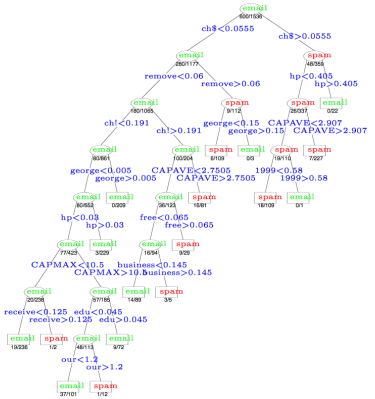
2. Classify algorithm: (a) If we use decision tree or Neural network, the result could be more accurate but will spend more time for calculating. (b) If we apply binary classification method, we may get more error because not all datasets could be binary classified.

3. Correlation of tweets and crime rate seems not so obvious as we expected. As we use positive and negative to classify each sentence, a more accurate rating process might give good results.

Demos

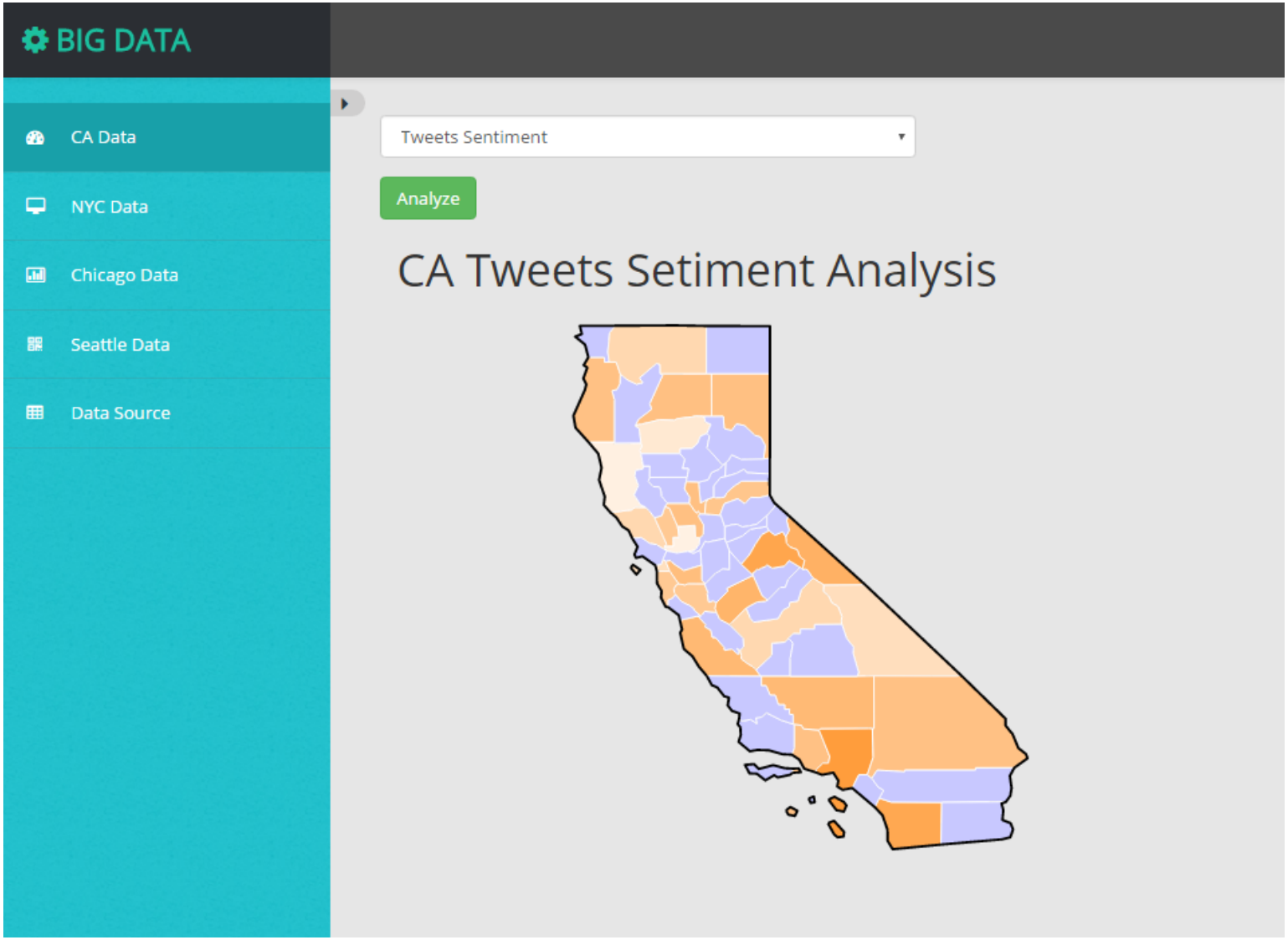
1. Feature extraction
2. Spark decision tree algorithm
3. Prediction.

```
"""
import pandas as pd
import numpy as np
from sklearn.cross_validation import KFold
from sklearn.feature_extraction.text import CountVectorizer
from pyspark.mllib.tree import DecisionTree, DecisionTreeModel
from pyspark.mllib.util import MLUtils
data = pd.read_csv("Sentiment.csv")
test=pd.read_csv("geoTweets.csv")
text=test["text"]
location=test["place__full_name"]
traindata=data['SentimentText']
train=[]
label=[]
a=0
trainlabel=data['Sentiment']
for i in range(0,len(traindata)):
    try :

        temp=traindata[i].encode('utf-8')
        train.append(temp.lower())
        label.append(trainlabel[i])
    except:
        a=a+1


ngram_vectorizer = CountVectorizer(analyzer='word', ngram_range=(1, 1), min_df=1)
traindata = ngram_vectorizer.fit_transform(train)
model = DecisionTree.trainClassifier(traindata, numClasses=2, categoricalFeaturesInfo={},impurity='gini', maxDepth=5, maxBins=32)
testdata=[]
testlocation=[]

for i in range(0,len(text[i])-1):
    if type(text) is str:
        print i
        print text[i].lower()
        testdata.append(text[i].lower())
        testlocation.append(location[i])

testdata=ngram_vectorizer.transform(testdata)
predictions = model.predict(testdata.map(lambda x: x.features))
```

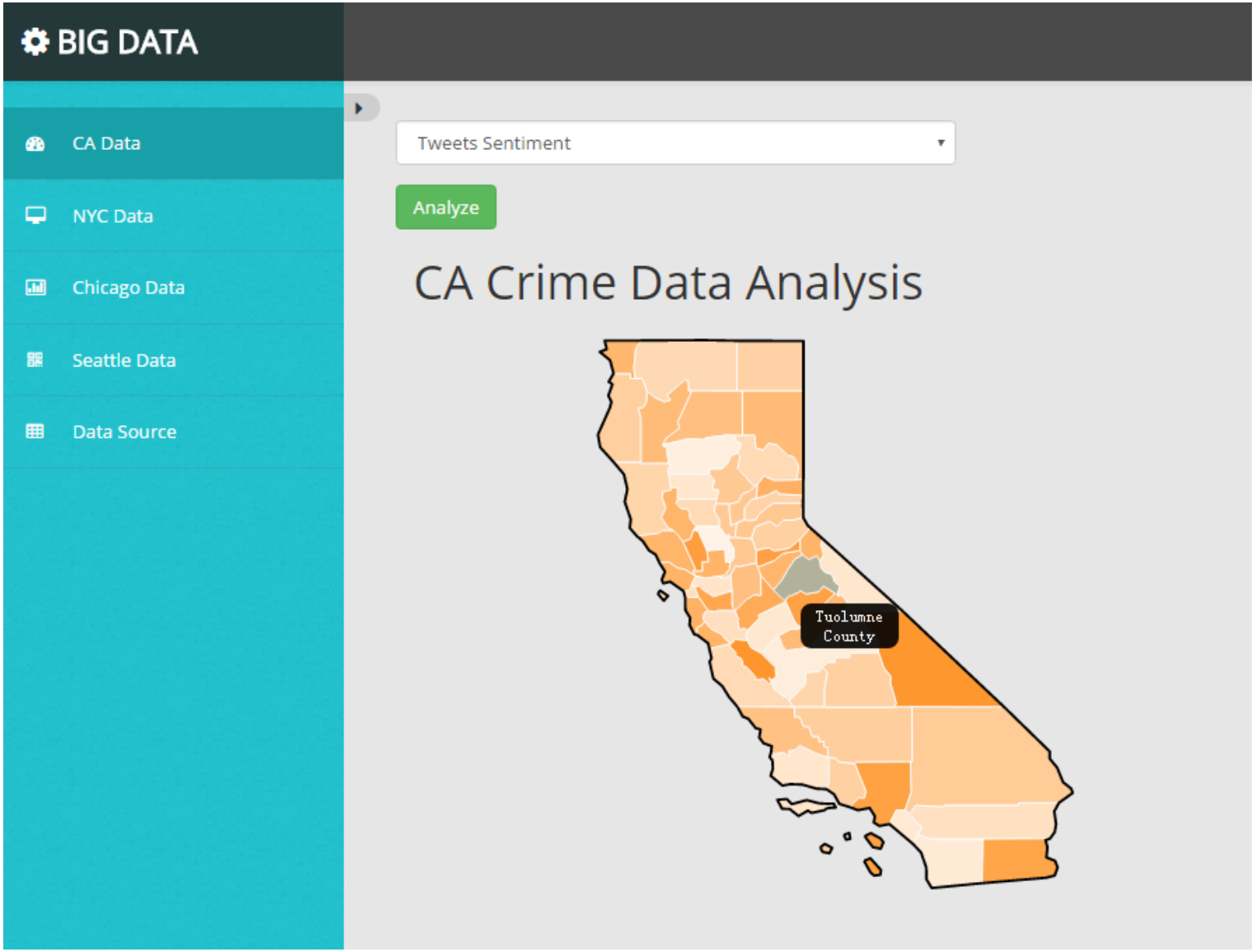


Bayonne, NJ	negative
Manhattan Beach, CA	neutral
Gardena, CA	negative
Tenafly, NJ	neutral
Fremont, CA	negative
Palos Heights, IL	neutral
Cambridge, MA	negative
Redwood City, CA	positive
Oakland, CA	neutral
Mountain View, CA	neutral
Hammond, IN	positive
Renton, WA	negative
Milpitas, CA	neutral
Hoboken, NJ	negative
Massachusetts, USA	neutral
Carson, CA	negative
Elmwood Park, IL	negative
Hackensack, NJ	neutral
Albany Park, Chicago	neutral
Downey, CA	negative
Santa Cruz, CA	negative
Oakland, CA	neutral
Franklin Park, IL	neutral
Carson, CA	negative
White Center, WA	neutral
Alhambra, CA	neutral
Newton, MA	neutral
Massachusetts, USA	neutral
New York, USA	positive

Demos



Demos



THANK YOU!