Topic 10

Text Analytics & Sentiment Analysis

Sentiment Analysis: Definition

- The computational field of study that analyzes people's opinions, sentiments, evaluations, appraisals, attitudes, and emotions **towards** entities such as **products, services, organizations, individuals, issues, events, topics**, and their **attributes** expressed in text (Liu, 2012).

Sentiment Analysis: Definition

- Using NLP, statistics, or machine learning methods to extract, identify, or otherwise characterize the sentiment content of a text unit
- Also known as *opinion mining*
- Represents a large problem space

What is Sentiment?

- An assumption on a binary opposition in opinions
- For/against, like/dislike, good/bad, etc.
- Some sentiment analysis jargons:
  - “Semantic orientation”
  - “Polarity”

Sentiment Analysis

- Sentiment analysis determines the attitude or inclination of a communicator through the contextual polarity of their speaking or writing
- May be based on :
  - ✔ own judgment
  - ✔ emotional state of subject
  - ✔ state of any emotional communication that affects a reader or listener
- Subjective impressions, not facts

Sentiment Analysis

- Tries to determine and classify a <u>person's state of mind on a subject matter</u>
- Information can be mined/extracted from texts, tweets, blogs, chats, social media sources, news articles or comments



Interesting Questions in SA

- Is this product review positive or negative?
- Is this customer email satisfied or dis-satisfied?
- Based on a sample of tweets, how are people responding to this ad campaign/product release/news item?
- How have bloggers' attitudes about the election?

Different Levels of Analysis

- **Document level**
  - Classify whether a whole opinion document expresses a positive or negative sentiment.
  - Assumes that each document expresses opinions on a single entity (e.g., a single product)
  - Example: product review

> I love this remote, it's easy to use, program, and the overall performance is excellent. I've used several universal remotes in the past, but this one's outstanding. Even the shape and design makes it comfortable for you to control.

Different Levels of Analysis

- **Sentence level**
  - Determines whether each sentence expressed a positive, negative, or neutral opinion.
  - Distinguishes sentences (objective sentences) that express factual information from sentences (subjective sentences) that express subjective views and opinions
  - Example:
    - "We bought the **<u>car</u>** last month and the windshield wiper has fallen off" (objective sentence + opinion)
    - "**<u>Apple</u>** is doing very well in this lousy economy"

Different Levels of Analysis

- **Entity and Aspect level**
  - Aspect level performs finer-grained analysis, also known as feature level
  - Directly looks at the opinion itself based on the idea that an opinion consists of a sentiment (positive or negative) and a target (of opinion)
  - Example: "The iPhone's call quality is good, but its battery life is short"
    - Evaluates two aspects, **<u>call quality</u>** and **<u>battery life</u>** of *iPhone* (entity)

Sentiment Analysis Lexicon

- There seems to be some relation between positive words and positive reviews
- We can come up with a set of keywords by hand to identify polarity
- Positive words: love, best, cool, great, good, amazing, helpful
- Negative words: hate, worst, awful, nightmare, fail
- Neutral words: unusual, confusing, not bad, perhaps, possible
  - **Neutral** usually means **no opinion** or words do not fall in either category

Customer Reviews on Airline Services (airline ratings.com & twitter)

Dear United baggage crew, please deliver my luggage. Let my luggage be found.

"Those #AmericanAirlines losers canceled my flight!"

Don't fly with @British_Airways. They can't keep track of your luggage.

I will **never** book with AirAsia X again. I had my scheduled flights **cancelled** on the 31st August and have been waiting ever since for my refund.

the reason y i ❤️ flying wth #MAS soo much.Delicious foods, superb entertainments and many more 😍 #malaysiaairlines

I fly a lot, but this is my first time on @VirginAmerica coolest plane I've ever been on. New new favorite airline!

I recently travelled on Malaysia Airlines from Sydney to KL. Everything was good and the service was excellent.

We had a very **bad experience** from **Qatar airways** and **crew**. we already complained to Qatar airways and the **reply** we got from them was really **disappointing**

Challenges in SA

- People express opinions in complex ways
- In opinion texts, lexical content alone can be misleading
- Intra-textual and sub-sentential reversals, negation, topic change are common
- Rhetorical devices/modes such as sarcasm, irony, implication, etc.

Steps in Sentiment Analysis using NLTK Manual Training/Testing

- Train classifier to automatically classify a tweet as a positive or negative tweet using NLTK.
- Need a list of manually classified tweets (training data: labelled) and unclassified tweets (test data: unlabelled). Examples:

Classified tweets (training)

Positive tweets

I like this laptop.
The view here is fantastic.
I feel great tonight.
I am very excited about the class project.
He is my best friend.

Negative tweets

I hate this laptop.
The view here is awful.
I feel tired tonight.
I am not looking forward to the class project.
He is my worst friend ever.

Unclassified tweets (testing)

Positive/Negative tweets?

I feel happy this morning.
Hussin is my friend.
I do not like that guy.
My car is not great.
Your tweet is annoying.

Steps in Sentiment Analysis using NLTK

- Import nltk
- Create two separate lists in Python for positive and negative tweets:

```
pos_tweets = [('I like this laptop', 'positive'),
              ('The food here is wonderful', 'positive'),
              ('I feel great tonight', 'positive'),
              ('I am very excited about the class project', 'positive'),
              ('He is my best friend', 'positive')]

neg_tweets = [('I hate this laptop', 'negative'),
              ('The food here is awful', 'negative'),
              ('I feel tired tonight', 'negative'),
              ('I am not looking forward to the class project', 'negative'),
              ('He is my worst friend ever', 'negative')]
```

Steps in Sentiment Analysis using NLTK

- Take both lists and create a single list of tuples each containing two elements (merge list into a tuple). First element is an array containing the words and second element is the type of sentiment.
- Set a property for extraction: Exclude words smaller than 2 characters and normalize tweets by setting all words in lowercase .

```
tweets = []
for (words, sentiment) in pos_tweets + neg_tweets:
   words_filtered = [e.lower() for e in words.split() if len(e) >= 3]
   tweets.append((words_filtered, sentiment))
```

Steps in Sentiment Analysis using NLTK

- Print the new merged list of tweets in the tuple

```
print(len(tweets))

for i in range(len(tweets)):
    print(tweets[i])
```

```
10
(['like', 'this', 'laptop'], 'positive')
(['the', 'food', 'here', 'wonderful'], 'positive')
(['feel', 'great', 'tonight'], 'positive')
(['very', 'excited', 'about', 'the', 'class', 'project'], 'positive')
(['best', 'friend'], 'positive')
(['hate', 'this', 'laptop'], 'negative')
(['the', 'food', 'here', 'awful'], 'negative')
(['feel', 'tired', 'tonight'], 'negative')
(['not', 'looking', 'forward', 'the', 'class', 'project'], 'negative')
(['worst', 'friend', 'ever'], 'negative')
```

Steps in Sentiment Analysis using NLTK

- Extract **word features** from the tweets (this is **Bag of Words** feature extraction method).
- This is a list with every distinct words ordered by frequency of appearance.
- Use the following function to get the list plus the two NLTK helper functions.

```
def get_words_in_tweets(tweets):
    all_words = []
    for (words, sentiment) in tweets:
        all_words.extend(words)
    return all_words

def get_word_features(wordlist):
    wordlist = nltk.FreqDist(wordlist)
    word_features = wordlist.keys()
    return word_features
```

Steps in Sentiment Analysis using NLTK

- Looking into variable 'wordlist' of the function get_word_features() :

```
>>> wordlist = nltk.FreqDist(get_words_in_tweets(tweets))
>>> print(wordlist)
<FreqDist with 25 samples and 37 outcomes>
>>> print(wordlist.most_common(25))
[('the', 4), ('friend', 2), ('here', 2), ('laptop', 2), ('class', 2)
, ('project', 2), ('tonight', 2), ('feel', 2), ('food', 2), ('this',
2), ('forward', 1), ('ever', 1), ('hate', 1), ('very', 1), ('worst',
1), ('tired', 1), ('best', 1), ('looking', 1), ('excited', 1), ('abo
ut', 1), ('great', 1), ('wonderful', 1), ('not', 1), ('awful', 1), (
'like', 1)]
```

- List of word_features:

```
>>> word_features = get_word_features(get_words_in_tweets(tweets))
>>> print(word_features)
dict_keys(['friend', 'here', 'laptop', 'forward', 'class', 'ever', '
project', 'the', 'hate', 'very', 'tonight', 'worst', 'tired', 'best'
, 'looking', 'excited', 'feel', 'about', 'great', 'wonderful', 'food
', 'not', 'this', 'awful', 'like'])
```

Steps in Sentiment Analysis using NLTK

- To create a classifier, we need to decide what features are relevant.
- We first need a feature extractor which returns a dictionary indicating what words are contained in the input passed (tweets).
- Use the previously extracted word features list with the input to create the dictionary.

```
def extract_features(document):
    document_words = set(document)
    features = {}
    for word in word_features:
        features['contains(%s)' % word] = (word in document_words)
    return features
```

Steps in Sentiment Analysis using NLTK

- Call the feature extractor with the document [‘like’, ‘this’, ‘laptop’] which is the first positive tweet.
- We obtain the following dictionary which indicates that the document contains the words: ‘like’, ‘this’ and ‘laptop’.
- This is a boolean word feature extractor

```
>>> extract_features(['like', 'this', 'laptop'])
{'contains(laptop)': True, 'contains(friend)': False, 'contains(not)': False,
'contains(here)': False, 'contains(the)': False, 'contains(forward)': False, '
contains(excited)': False, 'contains(food)': False, 'contains(looking)': False
, 'contains(project)': False, 'contains(feel)': False, 'contains(wonderful)':
False, 'contains(hate)': False, 'contains(this)': True, 'contains(like)': True
, 'contains(awful)': False, 'contains(tonight)': False, 'contains(class)': Fal
se, 'contains(very)': False, 'contains(about)': False, 'contains(great)': Fals
e, 'contains(ever)': False, 'contains(best)': False, 'contains(tired)': False,
'contains(worst)': False}
```

Training the Classifier with NLTK

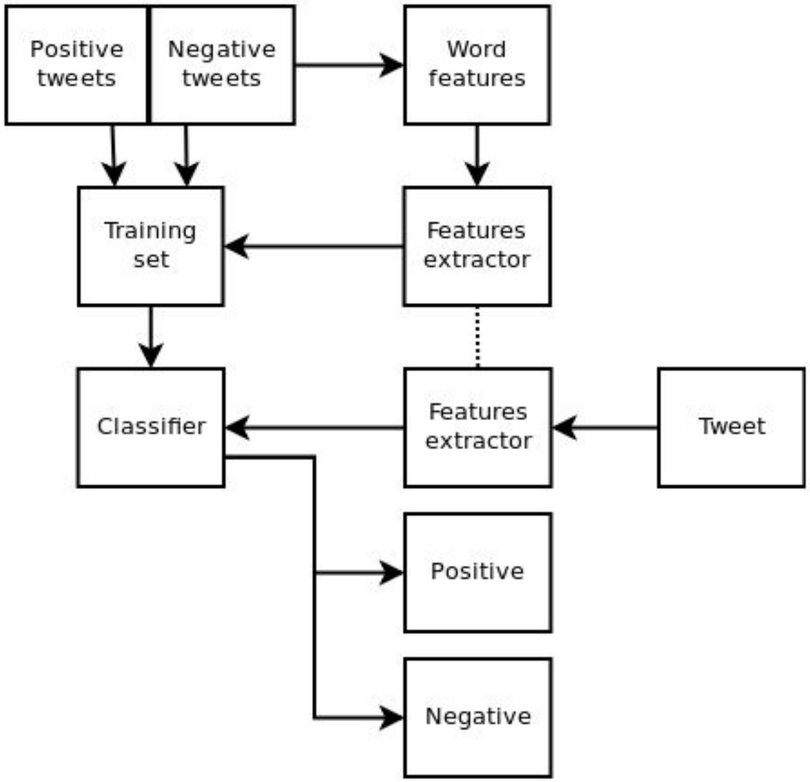
- Using our feature extractor, we apply the features to our classifier using the method apply_features()
- Pass the feature extractor along with our tweets list

```
>>> training_set = nltk.classify.apply_features(extract_features, tweets)
>>> print(training_set)
[({'contains(laptop)': True, 'contains(friend)': False, 'contains(not)': False
, 'contains(here)': False, 'contains(the)': False, 'contains(forward)': False,
'contains(excited)': False, 'contains(food)': False, 'contains(looking)': Fals
e, 'contains(project)': False, 'contains(feel)': False, 'contains(wonderful)':
False, 'contains(hate)': False, 'contains(this)': True, 'contains(like)': True
, 'contains(awful)': False, 'contains(tonight)': False, 'contains(class)': Fal
se, 'contains(very)': False, 'contains(about)': False, 'contains(great)': Fals
e, 'contains(ever)': False, 'contains(best)': False, 'contains(tired)': False,
'contains(worst)': False}, 'positive'), ({'contains(laptop)': False, 'contains
(friend)': False, 'contains(not)': False, 'contains(here)': True, 'contains(th
e)': True, 'contains(forward)': False, 'contains(excited)': False, 'contains(f
ood)': True, 'contains(looking)': False, 'contains(project)': False, 'contains
(feel)': False, 'contains(wonderful)': True, 'contains(hate)': False, 'contain
s(this)': False, 'contains(like)': False, 'contains(awful)': False, 'contains(
tonight)': False, 'contains(class)': False, 'contains(very)': False, 'contains
(about)': False, 'contains(great)': False, 'contains(ever)': False, 'contains(
best)': False, 'contains(tired)': False, 'contains(worst)': False}, 'positive'
), ...]
```

Training the Classifier with NLTK (cont.)

- We train our classifier with our training set:

```
>>> classifier = nltk.NaiveBayesClassifier.train(training_set)
```



Example for NLTK Movie Reviews

```
import nltk.classify.util
from nltk.classify import NaiveBayesClassifier
from nltk.corpus import movie_reviews

def word_feats(words):
    return dict([(word, True) for word in words])

negids = movie_reviews.fileids('neg')
posids = movie_reviews.fileids('pos')

negfeats = [(word_feats(movie_reviews.words(fileids=[f])), 'neg') for f in negids]
posfeats = [(word_feats(movie_reviews.words(fileids=[f])), 'pos') for f in posids]

negcutoff = len(negfeats)*3/4
poscutoff = len(posfeats)*3/4

trainfeats = negfeats[:negcutoff] + posfeats[:poscutoff]
testfeats = negfeats[negcutoff:] + posfeats[poscutoff:]
print 'train on %d instances, test on %d instances' % (len(trainfeats), len(testfeats))

classifier = NaiveBayesClassifier.train(trainfeats)
print 'accuracy:', nltk.classify.util.accuracy(classifier, testfeats)
classifier.show_most_informative_features()
```

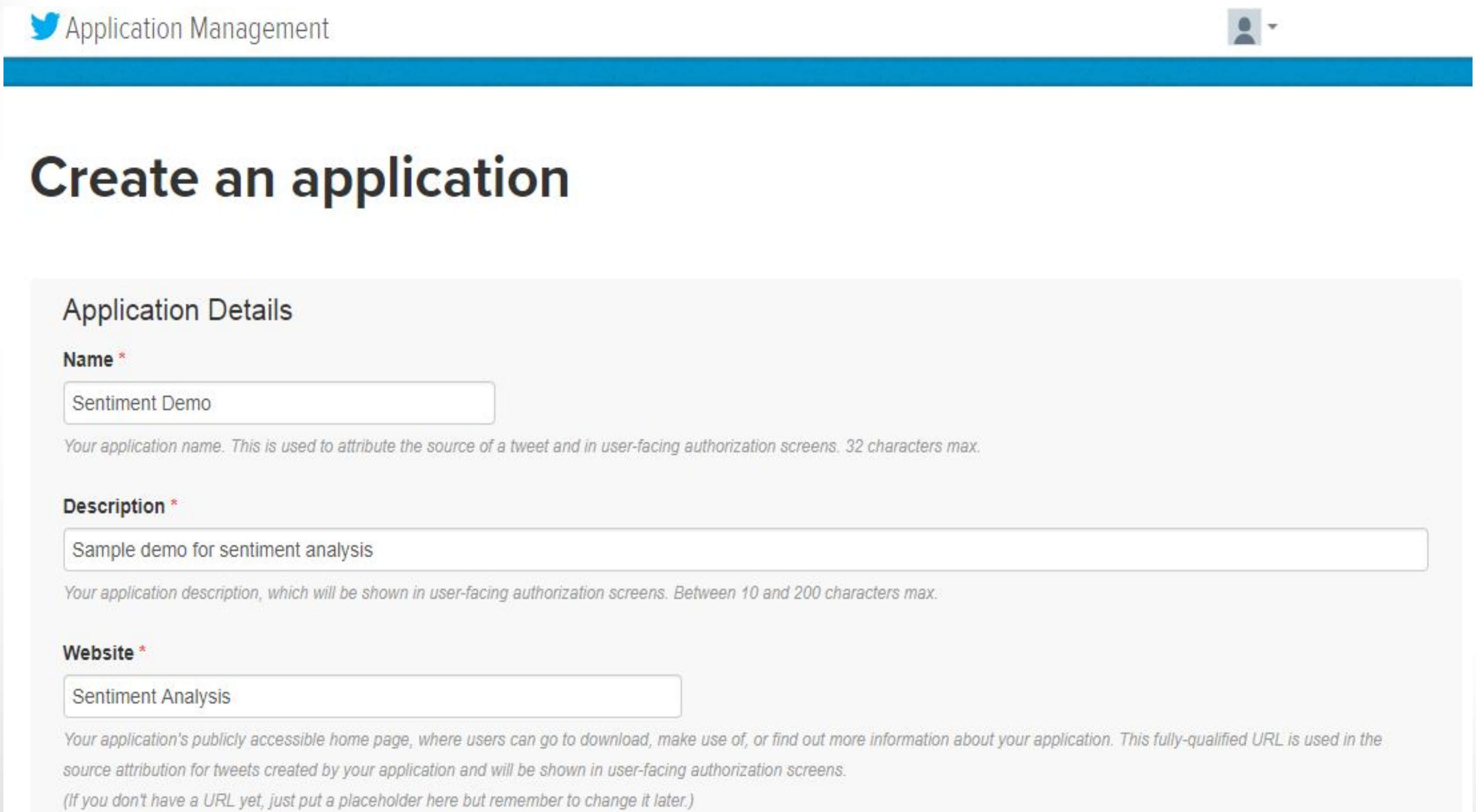
Text Analytics using Twitter data

Sentiment Analysis with Social Media Data: Twitter

- Steps:
  - Create a twitter account (if you do not have one)
  - Create a sample website. You need a URL to embed your twitter app (if you do not have one). Eg: blogspot, google site, wordpress, etc…
  - Create your twitter app by applying through the twitter developer page (refer next slides)

Creating New App with Twitter

- You need a twitter account
- Go to https://apps.twitter.com/app/new
- Create your application



Creating New App with Twitter (cont…)

Website *

Sentiment Analysis

Your application's publicly accessible home page, where users can go to download, make use of, or find out more information about your application. This fully-qualified URL is used in the source attribution for tweets created by your application and will be shown in user-facing authorization screens.
(If you don't have a URL yet, just put a placeholder here but remember to change it later.)

Callback URL

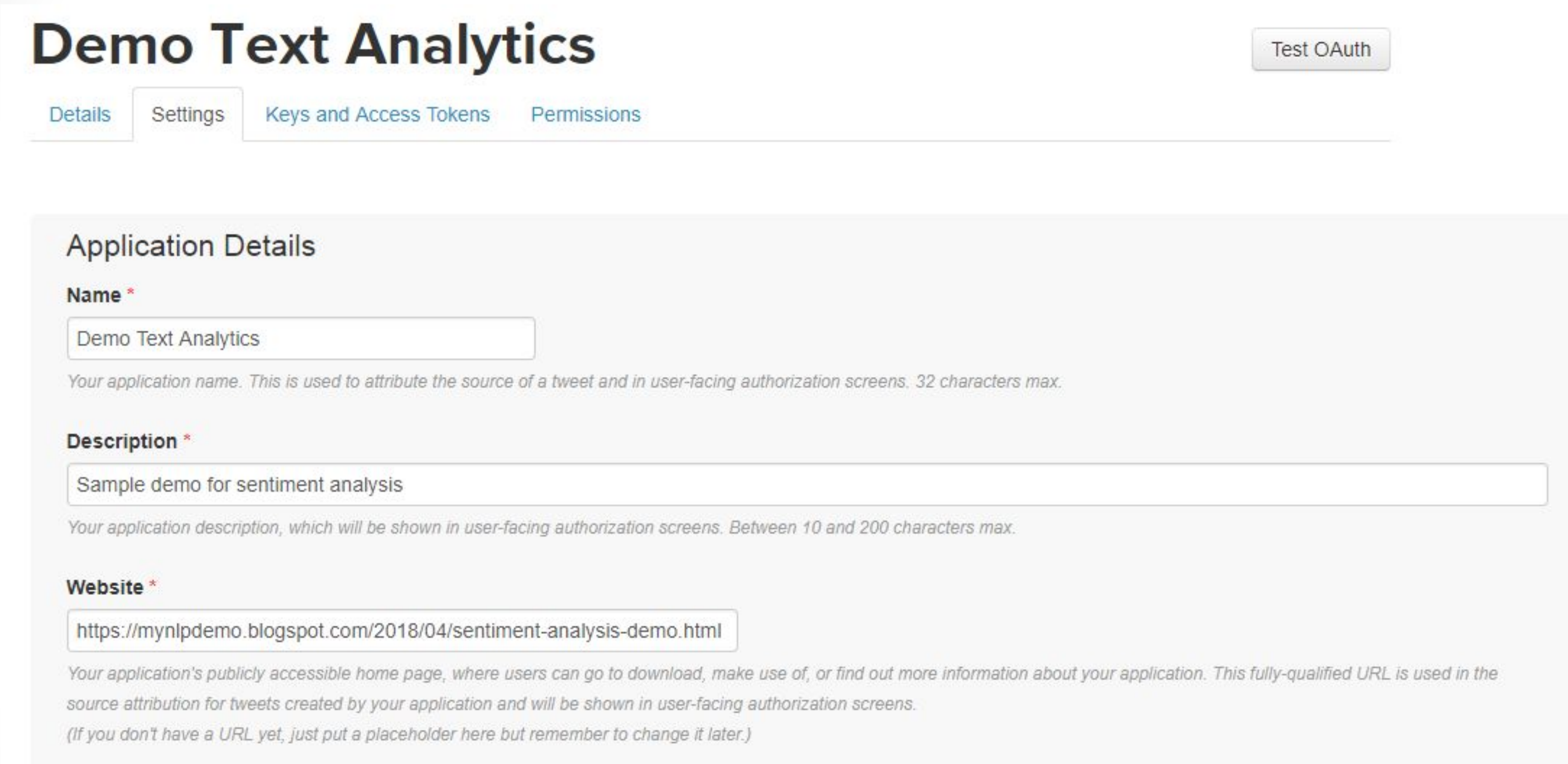
Where should we return after successfully authenticating? OAuth 1.0a applications should explicitly specify their oauth_callback URL on the request token step, regardless of the value given here. To restrict your application from using callbacks, leave this field blank.

Developer Agreement

☑ Yes, I have read and agree to the Twitter Developer Agreement.

Create your Twitter application

Creating New App with Twitter (cont…)



https://developer.twitter.com/en/docs/tweets/search/api-reference/get-search-tweets.html

Getting your API from twitter

Application Settings

Keep the "Consumer Secret" a secret. This key should never be human-readable in your application.

Consumer Key (API Key)	[redacted]
Consumer Secret (API Secret)	[redacted]
Access Level	Read and write (modify app permissions)
Owner	[redacted]
Owner ID	[redacted]

Application Actions

Regenerate Consumer Key and Secret   Change App Permissions

Your Access Token

This access token can be used to make API requests on your own account's behalf. Do not share your access token secret with anyone.

Access Token	[redacted]
Access Token Secret	[redacted]

Sentiment Analysis with Tweepy

```
import tweepy
import json
import re
import pandas as pd
import matplotlib.pyplot as plt
from tweepy import OAuthHandler
from tweepy import Stream
from tweepy.streaming import StreamListener
```

```
consumer_key = 'consumer_key'
consumer_secret = 'consumer_secret'
access_token = 'access_token'
access_secret = 'access_secret'
```

Replace these part with your own generated keys/tokens

```
#getting authorization from twitter with authentication properties
auth = OAuthHandler(consumer_key, consumer_secret)
auth.set_access_token(access_token, access_secret)

api = tweepy.API (auth)
file = open('Christchurch_shooting.dat','a')
```

Sentiment Analysis with Tweepy

```
class MyListener(StreamListener):

    def __init__(self, api=None):
        super(StreamListener, self).__init__()
        self.num_tweets = 0

    def on_data(self, data):

        try:
            with open('Christchurch_shooting.dat','a') as f:
                tweet = json.loads(data)
                if tweet["lang"]=="en":
                    file.write(data)
                    file.write("\n")
                if tweet["lang"]=="en" and tweet["user"]["location"]=="New Zealand":
                    if self.num_tweets < 2000:
                        print(json.dumps(tweet["text"], indent=4))
                        f.write(tweet["text"])
                        f.write("\n")
                        self.num_tweets += 1

                return True
        except BaseException as e:
            print("Error on_data: %s" % str(e))
        return True
```

Sentiment Analysis with Tweepy

```
    def on_error(self, status):
        print(status)
        return True

    def on_status(self, status):
        if status.retweeted_status:
           return
        print(status)

#online streaming of twitter data
mytwitter_stream = Stream(auth, MyListener())
#filter stream by keywords
mytwitter_stream.filter(track=['#NZMosqueShooting','#ChristchurchMosqueAttack','#ChristchurchShootings','Christchurch Shootings'])
file.close()

print("Done")
```

Sentiment Analysis Evaluation

- The use of sentiment lexicon/word list to determine polarity of sentiment (positive and negative words)
- Simple Summarization Score
  - Assign +1 to positive sentiment
  - Assign -1 to negative sentiment
  - Assign 0 to neutral sentiment
  - Accumulate score for each sentence:
    - Positive total = +ve sentiment
    - Negative total = -ve sentiment
    - Zero total = neutral sentiment

Pointwise Mutual Information (PMI)

- Information-theory approach to find collocations
  - Measure of how much one word tells us about the other. How much information we gain?
  - Can be negative or positive
  - Can be used with *n-gram* language model

Pointwise Mutual Information : Intuition

- Similar to Bayesian Modeling
  - Given two events, x’ and y’
    - In this case x’ and y’ are occurrences of particular words (a sentiment and some other words)
  - Denominator P(x) * P(y) is the expected value of P(x , y), assuming x and y are independent

Pointwise Mutual Information

$$
\begin{aligned}
I(x', y') &= \log_2 \frac{P(x'y')}{P(x')P(y')} \\
&= \log_2 \frac{P(x'|y')}{P(x')} \\
&= \log_2 \frac{P(y'|x')}{P(y')}
\end{aligned}
$$

Examples with PMI (Bigram)

Bigram	freq1	freq2	bigram freq	PMI
great britain	2	2	2	7.22
his assent	9	4	4	7.22
britain is	2	10	2	7.06
independent states	4	8	3	6.97
united states	3	8	2	6.8
human events	1	2	1	6.22
one people	1	10	1	6.22
equal station	2	1	1	6.22
mankind requires	3	1	1	6.22
becomes necessary	1	2	1	6.22
created equal	1	2	1	6.22
.....				
and the	57	78	3	-0.14
and our	57	26	1	-0.14
and of	57	79	3	-0.16
and for	57	29	1	-0.3
of and	79	57	1	-1.75

Examples with PMI (Bigram)

Bigram	freq1	freq2	bigram freq	PMI
my fellow	68	36	22	6.44
let us	95	220	69	6.38
fellow citizens	36	44	20	6.3
at home	78	22	19	6.23
on earth	141	30	14	5.6
one another	66	17	12	5.57
god bless	47	14	12	5.57
vice president	12	36	10	5.44
united states	21	18	12	5.3
both sides	20	13	8	5.1
....				
we and	644	1000	1	-4.67

Examples with PMI: Calculating Probabilities

```
# n_docs is the total n. of tweets
p_t = {}
p_t_com = defaultdict(lambda : defaultdict(int))

for term, n in count_stop_single.items():
    p_t[term] = n / n_docs
    for t2 in com[term]:
        p_t_com[term][t2] = com[term][t2] / n_docs


pmi = defaultdict(lambda : defaultdict(int))
for t1 in p_t:
    for t2 in com[t1]:
        denom = p_t[t1] * p_t[t2]
        pmi[t1][t2] = math.log2(p_t_com[t1][t2] / denom)
```

Problems with PMI

- Bad with sparse data
  - Words that only occur once, but appear together may get very high score PMI score
- High PMI score might not necessarily indicate importance of bigram

Other sources of Sentiment Analysis

- SentiWordNet
  - http://sentiwordnet.isti.cnr.it/
  - A lexical resource for opinion mining
  - Based on Wordnet synsets
  - Each synset is assigned three sentiment scores: positivity, negativity, and objectivity

Chat Analytics using Telegram data

https://towardsdatascience.com/introduction-to-the-telegram-api-b0cd220dbed2

https://medium.com/@jiayu./telegrammetry-stats-for-telegram-dcd075376f56