Understanding Twitter

In [10]:

```
import twitter
import simplejson as json
from __future__ import print_function
```

In [2]:

```
consumer_key = ''
consumer_secret = ''
access_token = ''
access_token_secret = ''
```

In [3]:

```
oauth = twitter.OAuth(access_token, access_token_secret, consumer_key, consumer_
secret)
t = twitter.Twitter(auth=oauth)
```

In [4]:

```
# Query the tweets - the result is a 'twitter.api.TwitterDictResponse' object
tweets_data = t.search.tweets(q='boogie cousins', result_type='recent', lang='en
', count=100)
# Convert the TwitterDictResponse object to a json file
tweets_json = json.dumps(tweets_data)
# Convert json file to a python dictionary
tweets = json.loads(tweets_json)
```

In [8]:

```
# Place all tweets into a list
tweets_list = []
for i in range(100):
    tweets_list.append(tweets['statuses'][i]['text'])
print('Number of tweets with repetition is ', len(tweets_list), '.', sep = '')
# To avoid repetition from retweets, remove duplicates
tweets_list = list(set(tweets_list))
print('Number of tweets without repetition is ', len(tweets_list), '.', sep='')
```

```
Number of tweets with repetition is 100.
Number of tweets without repetition is 73.
```

Popular Tweets

In [7]:

```
# Query the tweets - the result is a 'twitter.api.TwitterDictResponse' object
poptweets_data = t.search.tweets(q='boogie cousins', result_type='popular', lang
='en', count=100)
# Convert the TwitterDictResponse object to a json file
poptweets_json = json.dumps(poptweets_data)
# Convert json file to a python dictionary
poptweets = json.loads(poptweets_json)
```

In [199]:

```
# Place all tweets into a list
poptweets_list = []
for i in range(14):
    poptweets_list.append(poptweets['statuses'][i]['text'])
```

Understanding NLTK

Necessary Imports:

In [1]:

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

from nltk.classify import SklearnClassifier
from sklearn.svm import SVC
from sklearn.naive_bayes import BernoulliNB

from nltk.corpus import twitter_samples
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize
from nltk.corpus import wordnet
import re
from copy import deepcopy
```

Create Train and Test for Naive Bayes Classifier

Steps performed in this section:

1. Create a function that puts words into format that the classifier understands.
2. Load positive and negative tweets into respective lists, run them through step 1 function.
3. Make train and test set.
4. Train classifier.

In [10]:

```
tweets = twitter_samples.strings('negative_tweets.json')
tweets[:10]
```

Out[10]:

```
[u'hopeless for tmr :(',
 u"Everything in the kids section of IKEA is so cute. Shame I'm near
ly 19 in 2 months :(",
 u'@Hegelbon That heart sliding into the waste basket. :(',
 u'\u201c@ketchBurning: I hate Japanese call him "bani" :( :(\u201d\
n\nMe too',
 u'Dang starting next week I have "work" :(',
 u"oh god, my babies' faces :( https://t.co/9fcwGvaki0",
 u'@RileyMcDonough make me smile :((',
 u'@f0ggstar @stuartthull work neighbour on motors. Asked why and he
said hates the updates on search :( http://t.co/XvmTUikWln',
 u'why?:("@tahuodyy: sialan:( https://t.co/Hv1i0xcrL2"',
 u'Athabasca glacier was there in #1948 :-( #athabasca #glacier #jas
per #jaspernationalpark #alberta #explorealberta #\u2026 http://t.co
/dZZdqmf7Cz']
```

In [11]:

```
# Unsophisticated, piecewise version
def format_words(words):
    useful_words = [word for word in words if word not in stopwords.words("engli
sh")]
    my_dict = dict([(word, True) for word in useful_words])
    return(my_dict)

tweets = twitter_samples.strings('negative_tweets.json')
for i in range(len(tweets)):
        tweets[i] = tweets[i].replace(':', "").replace(')', '').replace('(', '')
.replace('\\', '').replace('/', '')
        tweets[i] = tweets[i].encode('ascii', 'ignore')
        tweets[i] = word_tokenize(tweets[i])

neg_tweets = []
[neg_tweets.append((format_words(tweet), 'negative')) for tweet in tweets]
print(neg_tweets[10])
```

```
({'good': True, 'g': True, '&': True, 'I': True, 'never': True, 'ide
a': True, 'going': True, 'meet': True, 'amp': True, ';': True, "'m":
True, 'really': True}, 'negative')
```

First Classifier

Step 1: Create Function

In [2]:

```
def format_words(tweets, tweet_type):
    # TWEET_TYPE MUST BE EITHER positive OR negative

    # First, clean the tweets (replace smileys, change format to string, tokeniz
e words)
    for i in range(len(tweets)):
        tweets[i] = tweets[i].replace(':', "").replace(')', '').replace('(', '')
.replace('\\', '').replace('/', '')
        tweets[i] = tweets[i].encode('ascii', 'ignore')
        tweets[i] = word_tokenize(tweets[i])

    # Create a list that will contain the dictionary of words in each tweet
    words = []
    for tweet in tweets:
        # Remove stopwords
        useful_words = [word for word in tweet if word not in stopwords.words('e
nglish')]
        # Create dictionary
        my_dict = dict([(word, True) for word in useful_words])
        # Append to words
        words.append((my_dict, tweet_type))
    # Return final list
    return(words)
```

Step 2: Load and clean data

In [3]:

```
negative = twitter_samples.strings('negative_tweets.json')
neg_tweets = format_words(negative, 'negative')
positive = twitter_samples.strings('positive_tweets.json')
pos_tweets = format_words(positive, 'positive')
```

Step 3: Create train and test set

In [14]:

```
type(neg_tweets[1])
```

Out[14]:

```
tuple
```

In [5]:

```
train = neg_tweets[1650:] + pos_tweets[1650:]
test = neg_tweets[:1650] + pos_tweets[:1650]
print(len(train),  len(test))
```

```
(6700, 3300)
```

Step 4: Train classifier

In [15]:

```
classifier = NaiveBayesClassifier.train(train)
accuracy = nltk.classify.util.accuracy(classifier, test)
print(accuracy * 100)
```

```
75.303030303
```

Step 5: Classifying an Example

In [16]:

```
example = tweets_list[4]
type(example)
print(example)
words = word_tokenize(example)
useful_words = [word for word in words if word not in stopwords.words('english')
]
exp = dict([(word, True) for word in words])
print(exp)
print('\n', classifier.classify(exp))
```

```
I liked a @YouTube video https://t.co/eyE7RQWVMM Boogie Cousins to G
olden State...WHAT?! | Strong Arm Sports Podcast Ep183
{'liked': True, '...': True, 'Cousins': True, 'Boogie': True, 'Sport
s': True, 'State': True, 'video': True, '!': True, 'WHAT': True, 'Yo
uTube': True, 'to': True, '//t.co/eyE7RQWVMM': True, 'https': True,
'Strong': True, '?': True, '@': True, 'I': True, 'Podcast': True, 'a
': True, 'Golden': True, 'Ep183': True, ':': True, '|': True, 'Arm':
True}

 positive
```

The above example is correctly identified as positive - Kevin Durant is welcoming Boogie. A very wholesome sentiment, indeed.

Second Classifier

Improvements to be made:

1. Remove mentions of usernames (@LebronJames)
2. Remove punctuation.
3. Remove retweet (RT).
4. Remove urls.

In [25]:

```
def format_words(tweets_input, tweet_type):
    # TWEET_TYPE MUST BE EITHER positive OR negative

    tweets = deepcopy(tweets_input)

    for i in range(len(tweets)):
        # Remove smileys
        tweets[i] = tweets[i].replace(':', "").replace(')', '').replace('(', '')
.replace('\\', '').replace('/', '')
        # Change format to strings
        tweets[i] = tweets[i].encode('ascii', 'ignore')
        # Remove usernames - Not helpful
        #tweets[i] = re.sub(r'@\S+', '', tweets[i])
        # Remove urls - Not helpful
        tweets[i] = re.sub(r'http\S+', '', tweets[i])
        # Remove puncutation
        tweets[i] = re.sub(r'[^\w\s]', '', tweets[i])
        # Remove indication of retweet (RT)
        tweets[i] = re.sub(r'RT\s', '', tweets[i])
        # Tokenize words
        tweets[i] = word_tokenize(tweets[i])

    # Create a list that will contain the dictionary of words in each tweet
    words = []
    for tweet in tweets:
        # Remove stopwords
        useful_words = [word for word in tweet if word not in stopwords.words('e
nglish')]
        # Create dictionary
        my_dict = dict([(word, True) for word in useful_words])
        # Append to words
        words.append((my_dict, tweet_type))
    # Return final list
    return(words)
```

In [26]:

```
negative = twitter_samples.strings('negative_tweets.json')
neg_tweets = format_words(negative, 'negative')
positive = twitter_samples.strings('positive_tweets.json')
pos_tweets = format_words(positive, 'positive')
```

In [27]:

```
train = neg_tweets[1650:] + pos_tweets[1650:]
test = neg_tweets[:1650] + pos_tweets[:1650]
print(len(train),  len(test))
classifier = NaiveBayesClassifier.train(train)
accuracy = nltk.classify.util.accuracy(classifier, test)
print(accuracy * 100)
```

```
6700 3300
76.5757575758
```

- Baseline - 75.303
- Removing urls and usernames reduce accuracy - 75.2.
- Just removing punctuation improves - 76.57.
- Doing all three - 76.18
- punctuation and usernames - 76.18
- punctuation and urls - 76.57

Conclusion: Remove urls and punctuation.

- punctuation, urls, removal of RT - 76.57.

Might as well remove RT since it'll be heavy in final data.

Using a Support Vector Classifier and Bernoulli Naive Bayes don't improve the accuracy - observe below:

In [33]:

```
classif = SklearnClassifier(SVC()).train(train)
print('SVM Accuracy:', nltk.classify.util.accuracy(classifier, test))

classif = SklearnClassifier(BernoulliNB()).train(train)
print('Bernoulli NB:', nltk.classify.util.accuracy(classifier, test))
```

```
SVM Accuracy: 0.765757575758
Bernoulli NB: 0.765757575758
```

Classifying Our Tweets

First, we need to redefine how our format_words function works since we don't want to give it a class. That was only for our training and testing.

In [34]:

```
def format_words_(tweets_input):

    # Create a copy of input list
    tweets = deepcopy(tweets_input)

    for i in range(len(tweets)):
        # Remove smileys
        tweets[i] = tweets[i].replace(':', "").replace(')', '').replace('(', '')
.replace('\\', '').replace('/', '')
        # Change format to strings
        tweets[i] = tweets[i].encode('ascii', 'ignore')
        # Remove usernames - Not helpful
        #tweets[i] = re.sub(r'@\S+', '', tweets[i])
        # Remove urls - Not helpful
        tweets[i] = re.sub(r'http\S+', '', tweets[i])
        # Remove puncutation
        tweets[i] = re.sub(r'[^\w\s]', '', tweets[i])
        # Remove indication of retweet (RT)
        tweets[i] = re.sub(r'RT\s', '', tweets[i])
        # Tokenize words
        tweets[i] = word_tokenize(tweets[i])

    # Create a list that will contain the dictionary of words in each tweet
    words = []
    for tweet in tweets:
        # Remove stopwords
        useful_words = [word for word in tweet if word not in stopwords.words('e
nglish')]
        # Create dictionary
        my_dict = dict([(word, True) for word in useful_words])
        # Append to words
        words.append(my_dict)
    # Return final list
    return(words)
```

In [35]:

```
input_data = format_words_(tweets_list)
```

In [36]:

```
pos = 0
neg = 0
for tweet in input_data:
    result = classifier.classify(tweet)
    if result == 'negative':
        neg = neg + 1
    if result == 'positive':
        pos = pos + 1
```

In [37]:

```
print('About {}% of tweets regarding Boogie Cousins were negative.'.format(round
((float(neg)/(neg+pos)*100), 2)))
```

```
About 23.29% of tweets regarding Boogie Cousins were negative.
```