In [11]:
```
#Make a model to determine whether a tweet positive or negative
import nltk
import string
from nltk.tag import pos_tag
from nltk.stem.wordnet import WordNetLemmatizer
from nltk import classify
from nltk.corpus import stopwords
from nltk import NaiveBayesClassifier
from random import shuffle
```

In [12]:
```
#Download the sample tweets
nltk.download('twitter_samples')
nltk.download('averaged_perceptron_tagger')
nltk.download('wordnet')
nltk.download('stopwords')
nltk.download('punkt')
nltk.download('omw-1.4')
```

```
[nltk_data] Downloading package twitter_samples to
[nltk_data]     /home/adel/nltk_data...
[nltk_data]   Package twitter_samples is already up-to-date!
[nltk_data] Downloading package averaged_perceptron_tagger to
[nltk_data]     /home/adel/nltk_data...
[nltk_data]   Package averaged_perceptron_tagger is already up-to-
[nltk_data]       date!
[nltk_data] Downloading package wordnet to /home/adel/nltk_data...
[nltk_data]   Package wordnet is already up-to-date!
[nltk_data] Downloading package stopwords to /home/adel/nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
[nltk_data] Downloading package punkt to /home/adel/nltk_data...
[nltk_data]   Package punkt is already up-to-date!
[nltk_data] Downloading package omw-1.4 to /home/adel/nltk_data...
[nltk_data]   Unzipping corpora/omw-1.4.zip.
```

Out[12]:
```
True
```

In [13]:
```
#Get the positive and negative tweets
positive_tweets = nltk.corpus.twitter_samples.strings('positive_tweets.json')
negative_tweets = nltk.corpus.twitter_samples.strings('negative_tweets.json')
```

In [14]:
```
positive_tweets[0]
```

Out[14]:
```
'#FollowFriday @France_Inte @PKuchly57 @Milipol_Paris for being top engaged members in my community this week :)'
```

In [15]:
```
#Tokenize the tweets
positive_tweets = nltk.corpus.twitter_samples.tokenized('positive_tweets.json')
negative_tweets = nltk.corpus.twitter_samples.tokenized('negative_tweets.json')
```

In [16]:
```
positive_tweets[0]
```

Out[16]:
```
['#FollowFriday',
 '@France_Inte',
 '@PKuchly57',
 '@Milipol_Paris',
 'for',
 'being',
 'top',
 'engaged',
 'members',
 'in',
 'my',
 'community',
 'this',
 'week',
 ':)']
```

In [17]:
```
#Remove noise from data
def is_clean(word: str):
  if word.startswith('@'):
    return False
  if word.startswith('http://') or word.startswith('https://'):
    return False
  if word in string.punctuation:
    return False
  if word.isnumeric():
    return False
  if word in stopwords.words('english'):
    return False
  return True


def clean_tokens(tokens: list):
  return [word.lower() for word in tokens if is_clean(word)]

positive_tweets_cleaned = [clean_tokens(tokens) for tokens in positive_tweets]
negative_tweets_cleaned = [clean_tokens(tokens) for tokens in negative_tweets]
```

In [18]:
```
positive_tweets_cleaned[0]
```

Out[18]:
```
['#followfriday', 'top', 'engaged', 'members', 'community', 'week', ':)']
```

In [19]:
```
negative_tweets_cleaned[0]
```

Out[19]:
```
['hopeless', 'tmr', ':(']
```

In [20]:
```
# Normalize the data
lemmatizer = WordNetLemmatizer()

 def  lemmatize(word: str, tag: str):
     if tag.startswith('NN'):
       pos = 'n'
     elif tag.startswith('VB'):
        pos = 'v'
     else:
        pos = 'a'
     return lemmatizer.lemmatize(word, pos)

def lemmatize_tokens(tokens:list):
  return [lemmatize(word, tag) for word, tag in pos_tag(tokens)]


positive_tweets_normalized = [lemmatize_tokens(tokens) for tokens in positive_tweets_cleaned]
negative_tweets_normalized = [lemmatize_tokens(tokens) for tokens in negative_tweets_cleaned]
```

In [21]:
```
positive_tweets_normalized[0]
```

Out[21]:
```
['#followfriday', 'top', 'engaged', 'member', 'community', 'week', ':)']
```

In [22]:
```
negative_tweets_normalized[0]
```

Out[22]:
```
['hopeless', 'tmr', ':(']
```

In [23]:
```
#Prepare data for Model
positive_dataset = [({token: True for token in tokens}, 'Positive') for tokens in positive_tweets_normalized]
negative_dataset = [({token: True for token in tokens}, 'Negative') for tokens in negative_tweets_normalized]
```

In [24]:
```
positive_dataset[0]
```

Out[24]:
```
({'#followfriday': True,
  'top': True,
  'engaged': True,
  'member': True,
  'community': True,
  'week': True,
  ':)': True},
 'Positive')
```

In [25]:
```
negative_dataset[0]
```

Out[25]:
```
({'hopeless': True, 'tmr': True, ':(': True}, 'Negative')
```

In [26]:
```
#Prepare training and test dataset
dataset = positive_dataset + negative_dataset

shuffle(dataset)

train_ds = dataset[:7000]
test_ds = dataset[7000:]
```

In [ ]:

In [27]:
```
#Train and test Model
classifier = NaiveBayesClassifier.train(train_ds)
```

In [28]:
```
classify.accuracy(classifier, test_ds)
```

Out[28]:
```
0.9963333333333333
```

In [29]:
```
#Show the most informative features
classifier.show_most_informative_features(10)
```

```
Most Informative Features
                      :) = True           Positi : Negati =    991.9 : 1.0
                     sad = True           Negati : Positi =     25.0 : 1.0
                     bam = True           Positi : Negati =     22.4 : 1.0
                follower = True           Positi : Negati =     19.8 : 1.0
                     too = True           Negati : Positi =     19.0 : 1.0
              appreciate = True           Positi : Negati =     17.0 : 1.0
                     x15 = True           Negati : Positi =     17.0 : 1.0
               community = True           Positi : Negati =     16.4 : 1.0
                    damn = True           Negati : Positi =     14.3 : 1.0
                  arrive = True           Positi : Negati =     12.2 : 1.0
```

In [30]:
```
#Test the model
tweet = 'this is fun and awesome'
tweet_dict = {token: True for token in lemmatize_tokens(clean_tokens(tweet.split()))}
```

In [31]:
```
classifier.classify(tweet_dict)
```

Out[31]:
```
'Positive'
```

Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js