In [102]:
```python
# Use tweet API to script the data
import csv
import pandas as pd
import json
####input your credentials here
consumer_key = 'HC3mhnbHYUNnS1Zfwnuv2C8dO'
consumer_secret = '9PEqSawfQSGKXytX5IJr4flOo0Bkp42h0zlXZc9jdzmDmR6fm3'
access_token = '1193698595658240000-7noD2IYnnoKPICU4LFtZLo7Ja9IUwz'
access_token_secret = 'oNeYKX2Bql8DWBOusAAPuM2N6qFFoXVyN6OQ4CrcegmRE'
from tweepy import API
from tweepy import Cursor
from tweepy.streaming import StreamListener
from tweepy import OAuthHandler
from tweepy import Stream


import numpy as np
import pandas as pd
import re
import nltk
from nltk.corpus import stopwords
```

Q3 (Bonus): Biterm Topic Model (BTM)

- There are many variants of LDA model. BTM is one designed for short text, while lDA in general expects documents with rich content.
- Read this paper carefully http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.402.4032&rep=rep1&type=pdf and try to understand the design
- Try the following experiments:
  - Script a few thousand tweets by different hastags
  - Run LDA and BTM respectively to discover topics among the collected tweets. BTM package can be found at https://pypi.org/project/biterm/
  - Compare the performance of each model. If one model works better, explain why it works better,
- Summarize your experiment in a pdf document.
- Note there is no absolute right or wrong answer in this experiment. All you need is to give a try and understand how BTM works and differences between BTM and LDA

1) : Script a few thousand tweets by different hastags

- Use the tweepy to script tweets by such different hastags:
  - Politics
  - Sports
  - Entertainment

In [103]:
```python
# # # # TWITTER AUTHENTICATER # # # #
class TwitterAuthenticator():
    def authenticate_twitter_app(self):
        auth = OAuthHandler(consumer_key, consumer_secret)
        auth.set_access_token(access_token,access_token_secret)
        return auth
# # # # TWITTER STREAMER # # # #
class TwitterStreamer():
    # Class for streaming and processing live tweets.
    def __init__(self):
        self.twitter_autenticator = TwitterAuthenticator()

    def stream_tweets(self, fetched_tweets_filename, hash_tag_list):
        # This handles Twitter authetification and the connection to Twitter Streaming API
        listener = TwitterListener(fetched_tweets_filename)
        auth = self.twitter_autenticator.authenticate_twitter_app()
        stream = Stream(auth, listener)
        # This line filter Twitter Streams to capture data by the keywords:
        stream.filter(track=hash_tag_list)
# # # # TWITTER STREAM LISTENER # # # #
class TwitterListener(StreamListener):
    # This is a basic listener that just prints received tweets to stdout.
    def __init__(self, fetched_tweets_filename):
        self.fetched_tweets_filename = fetched_tweets_filename

    def on_data(self, data):
        try:
            print(data)
            with open(self.fetched_tweets_filename, 'a') as tf:
                tf.write(data)
            return True
        except BaseException as e:
            print("Error on_data %s" % str(e))
        return True

    def on_error(self, status):
        if status == 420:
            # Returning False on_data method in case rate limit occurs.
            return False
        print(status)
```

In [104]:
```
# Script tweets with POLITICS hash_tag
# Store the data into a json file
hash_tag_list_political = ['parties','policy','usa','republican',
                           'political','Whitehouse','party','government',
                           'politics','american']
fetched_tweets_filename_political = "tweets_political.json"

twitter_streamer_political = TwitterStreamer()
twitter_streamer_political.stream_tweets(fetched_tweets_filename_political, hash_tag_list_political)
```

In [105]:
```
# Script tweets with Sports hash_tag
# Store the data into a json file
hash_tag_list_sport = ['sport','fitness','training','gym','motivation',
                       'sports','fit','workout','football']
fetched_tweets_filename_sport = "tweets_sport.json"

twitter_streamer_sport = TwitterStreamer()
twitter_streamer_sport.stream_tweets(fetched_tweets_filename_sport, hash_tag_list_sport)
```

In [106]:
```
# Script tweets with entertainment hash_tag
# Store the data into a json file
hash_tag_list_entertainment = ['entertainment','music','fun','hiphop',
                               'comedy','events','funny','dance']
fetched_tweets_filename_entertainment = "tweets_entertainment.json"

twitter_streamer_entertainment = TwitterStreamer()
twitter_streamer_entertainment.stream_tweets(fetched_tweets_filename_entertainment, hash_tag_list_entertainment)
```

In [135]:
```
# After Script these three different hash_tag data, we put them in the list:
tweet_political = []
tweet_sport = []
tweet_entertainment = []
for line in open('tweets_political.json', 'r'):
    tweet_political.append(json.loads(line))

for line in open('tweets_sport.json', 'r'):
    tweet_sport.append(json.loads(line))

for line in open('tweets_entertainment.json', 'r'):
    tweet_entertainment.append(json.loads(line))
```

In [169]:
```
# Clean the data
# What we need is english word
# Not punctuation or emoji
def tokenize(text):

    tokens = None
    text_lower = text.lower()
    stop_words = set(stopwords.words('english'))
    pattern = r'[a-zA-Z]+[-\._\']*[a-zA-Z]+'
    token1 = nltk.regexp_tokenize(text_lower,pattern)
    stop_extent = ['https','http']
    tokens = [i for i in token1 if i not in stop_words and stop_extent]
    try:
        tokens.remove('https')
        return tokens
    except:
        return tokens

def to_text(lis):
    li = []
    for i in lis:
        try:
            token = tokenize(i['text'])

            if token[0] == 'rt':
                token.remove('rt')
                li.append(' '.join(token))
            else:
                li.append(' '.join(token))
        except:
            continue
    return li
```

In [170]:
```
pol_text = to_text(tweet_political)
spo_text = to_text(tweet_sport)
ent_text = to_text(tweet_entertainment)
print('Number of tweet about political is  %d' % len(pol_text))
print('Number of tweet about sport is %d' % len(spo_text))
print('Number of tweet about entertainment is %d' % len(ent_text))
```

```
Number of tweet about political is  996
Number of tweet about sport is 455
Number of tweet about entertainment is 1230
```

In [171]:
```
# Label the data and put it in a DataFrame
pol_df = pd.DataFrame(pol_text,columns = ['text'])
pol_df['label'] = ['politics']*len(pol_text)

spo_df = pd.DataFrame(spo_text,columns = ['text'])
spo_df['label'] = ['sports']*len(spo_text)

ent_df = pd.DataFrame(ent_text,columns = ['text'])
ent_df['label'] = ['entertainment']*len(ent_text)
```

In [172]:
```
# Show the data
# The data of politics tweets
pol_df.head(4)
```

Out[172]:

	text	label
0	tedlieu served active duty never imagined russ...	politics
1	hsheilley transylvaniawvb block party	politics
2	ironhorse time let's get party started follow ...	politics
3	flwendywilliams excuse sunshine state leading ...	politics

In [173]:
```
# The data of sports tweets
spo_df.head(4)
```

Out[173]:

	text	label
0	i'm cleared workout i'll starting tomorrow i'm...	sports
1	hartsville vs beaufort south carolina high sch...	sports
2	yoooooo always pull planet fitness crying like	sports
3	annaelzalalisa yg thai blackpink	sports

In [153]:
```
# The data of entertainment tweets
ent_df.head(3)
```

Out[153]:

	text	label
0	pdf download jokes riddles kids know get daily...	entertainment
1	costa feat bebe ficci prod mygal official musi...	entertainment
2	adultmoodz forever funny t.co trze	entertainment

In [174]:
```
# Put these data set together
data = pd.concat([pol_df,spo_df,ent_df])
# data.to_csv('tweet_data.csv')
data
```

Out[174]:

	text	label
0	tedlieu served active duty never imagined russ...	politics
1	hsheilley transylvaniawvb block party	politics
2	ironhorse time let's get party started follow ...	politics
3	flwendywilliams excuse sunshine state leading ...	politics
4	rising_serpent schiff lied mean whistleblower ...	politics
...	...	...
1225	jesusandres oh nah absolutely straight heat mu...	entertainment
1226	scoulios ari lennox makes music people try get...	entertainment
1227	choi_bts jm heard first thought new year song ...	entertainment
1228	arilennox unacceptable t.co qfvdiosvab	entertainment
1229	ahappybri messy tried fun toile t.co jvaf btsa	entertainment

2681 rows × 2 columns

2) : Run LDA and BTM respectively to discover topics among the collected tweets.

2.1 Use LDA to discover topics

In [175]:
```
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.model_selection import train_test_split
tf_vectorizer = CountVectorizer(max_df=0.60,min_df=5, stop_words='english')
tf = tf_vectorizer.fit_transform(data['text'])
tf_feature_names = tf_vectorizer.get_feature_names()
# split dataset into train (90%) and test sets (10%)
# the test sets will be used to evaluate proplexity of topic modeling
X_train, X_test = train_test_split(tf, test_size=0.1, random_state=0)
```

In [176]:
```
X_test
```

Out[176]:
```
<269x985 sparse matrix of type '<class 'numpy.int64'>'
	with 1345 stored elements in Compressed Sparse Row format>
```

In [177]:
```
# LDA to discover topics
from sklearn.decomposition import LatentDirichletAllocation
num_topics = 3
# Run LDA
# max_iter control the number of iterations
# evaluate_every determines how often the perplexity is calculated
# n_jobs is the number of parallel threads
lda = LatentDirichletAllocation(n_components=num_topics, \
                                max_iter=25,verbose=1,
                                evaluate_every=1, n_jobs=1,
                                random_state=0).fit(X_train)
```

```
iteration: 1 of max_iter: 25, perplexity: 800.6773
iteration: 2 of max_iter: 25, perplexity: 728.4888
iteration: 3 of max_iter: 25, perplexity: 701.2713
iteration: 4 of max_iter: 25, perplexity: 687.4196
iteration: 5 of max_iter: 25, perplexity: 678.4874
iteration: 6 of max_iter: 25, perplexity: 671.5948
iteration: 7 of max_iter: 25, perplexity: 667.1603
iteration: 8 of max_iter: 25, perplexity: 664.0237
iteration: 9 of max_iter: 25, perplexity: 661.2024
iteration: 10 of max_iter: 25, perplexity: 658.7694
iteration: 11 of max_iter: 25, perplexity: 656.1627
iteration: 12 of max_iter: 25, perplexity: 653.6777
iteration: 13 of max_iter: 25, perplexity: 651.5920
iteration: 14 of max_iter: 25, perplexity: 650.0327
iteration: 15 of max_iter: 25, perplexity: 648.9127
iteration: 16 of max_iter: 25, perplexity: 647.9489
iteration: 17 of max_iter: 25, perplexity: 647.1541
iteration: 18 of max_iter: 25, perplexity: 645.9048
iteration: 19 of max_iter: 25, perplexity: 645.0392
iteration: 20 of max_iter: 25, perplexity: 644.5297
iteration: 21 of max_iter: 25, perplexity: 644.1466
iteration: 22 of max_iter: 25, perplexity: 643.7676
iteration: 23 of max_iter: 25, perplexity: 643.3231
iteration: 24 of max_iter: 25, perplexity: 642.7123
iteration: 25 of max_iter: 25, perplexity: 641.9569
```

In [178]:
```
# Check topic and word distribution per topic

num_top_words = 5

# lda.components_ returns a KxN matrix
# for word distribution in each topic.
# Each row consists of
# probability (counts) of each word in the feature space

for topic_idx, topic in enumerate(lda.components_):
    print ("Topic %d:" % (topic_idx))
    # print out top 20 words per topic
    words=[(tf_feature_names[i],topic[i]) \
           for i in topic.argsort()[::-1][0:num_top_words]]
    print(words)
    print("\n")
```

```
Topic 0:
[('football', 144.2870532707118), ('music', 136.86626300830164), ('vs', 101.66274335494305), ('high', 94.3228916172
0732), ('school', 92.31780522691463)]


Topic 1:
[('fun', 192.30786914855716), ('music', 131.6782599242316), ('funny', 115.2961034386048), ('party', 105.28317938111
047), ('like', 82.45053510185619)]


Topic 2:
[('people', 107.26309865473243), ('bts', 95.30058897729322), ('american', 80.28479999409035), ('trump', 71.41274451
252774), ('dance', 61.38922363789177)]
```

As we know there are three topics:

- Politics
- Sports
- Entertainment

From the result of the LDA:

- The first topic contains 'football', 'music', 'vs' the most. So it may be attributed to the sports sector. This is reasonable, because 'football' is a sports word and 'vs' may appear anywhere like Team A vs Team B.
- The second topic contains 'fun', 'music', 'funny' the most. So it may be attributed to the Entertainment sector.These three word are all relative to Entertainment.
- The third topic contains 'people', 'american', 'trump' the most. So it may be attributed to the politics.

So far the performance of LDA is good. We use it to discover three topics as we expecting.

2.2 Use BTM to discover topics

In [179]:
```
import numpy as np
import pyLDAvis
from biterm.cbtm import oBTM
from sklearn.feature_extraction.text import CountVectorizer
from biterm.utility import vec_to_biterms, topic_summuary # helper functions

if __name__ == "__main__":

    texts = data['text']

    # vectorize texts
    vec = CountVectorizer(stop_words='english')
    X = vec.fit_transform(texts).toarray()

    # get vocabulary
    vocab = np.array(vec.get_feature_names())

    # get biterms
    biterms = vec_to_biterms(X)

    # create btm
    btm = oBTM(num_topics=3, V=vocab)

    print("\n\n Train Online BTM ..")
    for i in range(0, len(biterms), 100): # prozess chunk of 200 texts
        biterms_chunk = biterms[i:i + 100]
        btm.fit(biterms_chunk, iterations=50)
    topics = btm.transform(biterms)

    print("\n\n Topic coherence ..")
    topic_summuary(btm.phi_wz.T, X, vocab, 10)

    #print("\n\n Texts & Topics ..")
    #for i in range(len(texts)):
        #print("{} (topic: {})".format(texts[i], topics[i].argmax()))
```

In [181]:
```
print("\n\n Topic coherence ..")
topic_summuary(btm.phi_wz.T, X, vocab, 5)
```

```
 Topic coherence ..
Topic 0 | Coherence=-36.56 | Top words= music fun football vs school
Topic 1 | Coherence=-42.22 | Top words= music dance new party funny
Topic 2 | Coherence=-1.88 | Top words= immigration aoc white supremacist controlling
```

Out[181]:
```
{'coherence': [-36.56375158868816, -42.222560495687134, -1.8832251965670652],
 'top_words': [array(['music', 'fun', 'football', 'vs', 'school'], dtype='<U50'),
  array(['music', 'dance', 'new', 'party', 'funny'], dtype='<U50'),
  array(['immigration', 'aoc', 'white', 'supremacist', 'controlling'],
        dtype='<U50')]}
```

From the result of BTM:

- For the topic 0, the similar thing is it contains 'football', 'vs' which indicate this is the sports topic
- For the topic 1, 'music', 'dance', 'funny' these words indicate this is the Entertainment topic.
- For the topic 2, words like 'immigration', 'white', 'supremacist' strongly indicate this topic is about Politics

The result of the BTM is also good.

3) : Compare the performance of each model

- So far they both did well to discover the these three topics.Actually the LDA performs better because there are some disturbing term in the BTM result like 'music' both appear in topic 0 and topic 1
- The difference between these two model is that BTM is more suitable for the short text. This tweeter example is about short text.Althought it may be unexpected, it is reasonalbe because of some random factors