- 学号：16340181
- 姓名：彭伟林

Part1 Bag of Words

三个部分的运行结果保存在Bag_of_Words_model.csv, Word2Vec_AverageVectors.csv, BagOfCentroids.csv三个文件当中。

由于代码中存在一些弃用的方法，我们添加一下代码可以使得这部分不进行输出，让我们观察结果的时候更加清晰。

In [24]:

```
import warnings
warnings.filterwarnings('ignore')
```

引入需要使用的包

In [64]:

```
import os
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.ensemble import RandomForestClassifier
import pandas as pd
import numpy as np
import re
import nltk
from nltk.corpus import stopwords
from gensim.models import Word2Vec
from bs4 import BeautifulSoup
from sklearn.cluster import KMeans
import time
```

为了处理的方便，首先我们要对评论的文本进行一些预处理的工作，包括去掉评论中存在的html标签，去掉一些非字母元素，去掉停用词；做完上面的预处理以后，一个整体的评论被分割成单个的单词，所以最后我们需要将单词拼成一个整体。

In [30]:

```
def review_to_words( raw_review ):
    # Function to convert a raw review to a string of words
    # The input is a single string (a raw movie review), and
    # the output is a single string (a preprocessed movie review)
    #
    # 1. Remove HTML
    review_text = BeautifulSoup(raw_review).get_text()
    #
    # 2. Remove non-letters
    letters_only = re.sub("[^a-zA-Z]", " ", review_text)
    #
    # 3. Convert to lower case, split into individual words
    words = letters_only.lower().split()
    #
    # 4. In Python, searching a set is much faster than searching
    #   a list, so convert the stop words to a set
    stops = set(stopwords.words("english"))
    #
    # 5. Remove stop words
    meaningful_words = [w for w in words if not w in stops]
    #
    # 6. Join the words back into one string separated by space,
    # and return the result.
    return( " ".join( meaningful_words ))
```

从tsv文件当中去读训练数据，这里是有监督训练，所以我们需要有标签的数据集。

In [31]:

```
train = pd.read_csv("./data/labeledTrainData.tsv", header=0, \
                    delimiter="\t", quoting=3)
```

将读取出来的句子进行处理，评论之间需要分开处理，预处理完成以后将评论存储在一个list中。

In [32]:

```
# Get the number of reviews based on the dataframe column size
num_reviews = train["review"].size

# Initialize an empty list to hold the clean reviews
clean_train_reviews = []
for i in range( 0, num_reviews ):
    # If the index is evenly divisible by 1000, print a message
    if( (i+1)%1000 == 0 ):
        print("Review %d of %d\n" % ( i+1, num_reviews ))

    clean_train_reviews.append( review_to_words( train["review"][i] ))
```

```
Review 1000 of 25000

Review 2000 of 25000

Review 3000 of 25000

Review 4000 of 25000

Review 5000 of 25000

Review 6000 of 25000

Review 7000 of 25000

Review 8000 of 25000

Review 9000 of 25000

Review 10000 of 25000

Review 11000 of 25000

Review 12000 of 25000

Review 13000 of 25000

Review 14000 of 25000

Review 15000 of 25000

Review 16000 of 25000

Review 17000 of 25000

Review 18000 of 25000

Review 19000 of 25000

Review 20000 of 25000

Review 21000 of 25000

Review 22000 of 25000

Review 23000 of 25000

Review 24000 of 25000

Review 25000 of 25000
```

按照Bag of Words的方法将评论转换成向量，为了节省空间与训练的时间，我们选取最常用的5000个单词作为词包,也就是说，每条评论被转换为一个size为5000的向量。

In [33]:

```
# Initialize the "CountVectorizer" object, which is scikit-learn's
# bag of words tool.
vectorizer = CountVectorizer(analyzer = "word",   \
                             tokenizer = None,    \
                             preprocessor = None, \
                             stop_words = None,   \
                             max_features = 5000)

# fit_transform() does two functions: First, it fits the model
# and learns the vocabulary; second, it transforms our training data
# into feature vectors. The input to fit_transform should be a list of
# strings.
train_data_features = vectorizer.fit_transform(clean_train_reviews)

# Numpy arrays are easy to work with, so convert the result to an
# array
train_data_features = train_data_features.toarray()
```

使用随机森林的方法进行训练，使用的树的个数为10。

In [35]:

```
# Initialize a Random Forest classifier with 100 trees
forest = RandomForestClassifier(n_estimators = 100)

# Fit the forest to the training set, using the bag of words as
# features and the sentiment labels as the response variable
#
# This may take a few minutes to run
forest = forest.fit( train_data_features, train["sentiment"] )
```

对测试文件当中的评论的情感进行预测并将结果输出到文件当中，注意是这里的评论也要有与训练集一样的预处理操作。

In [36]:

```
# Read the test data
test = pd.read_csv("./data/testData.tsv", header=0, delimiter="\t", \
                   quoting=3 )

# Verify that there are 25,000 rows and 2 columns
print(test.shape)

# Create an empty list and append the clean reviews one by one
num_reviews = len(test["review"])
clean_test_reviews = []

print("Cleaning and parsing the test set movie reviews...\n")
for i in range(0,num_reviews):
    if( (i+1) % 1000 == 0 ):
        print("Review %d of %d\n" % (i+1, num_reviews))
    clean_review = review_to_words( test["review"][i] )
    clean_test_reviews.append( clean_review )

# Get a bag of words for the test set, and convert to a numpy array
test_data_features = vectorizer.transform(clean_test_reviews)
test_data_features = test_data_features.toarray()

# Use the random forest to make sentiment label predictions
result = forest.predict(test_data_features)

# Copy the results to a pandas dataframe with an "id" column and
# a "sentiment" column
output = pd.DataFrame( data={"id":test["id"], "sentiment":result} )

# Use pandas to write the comma-separated output file
output.to_csv( "Bag_of_Words_model.csv", index=False, quoting=3 )
```

```
(25000, 2)
Cleaning and parsing the test set movie reviews...

Review 1000 of 25000
```

```
Review 1000 of 25000

Review 2000 of 25000

Review 3000 of 25000

Review 4000 of 25000

Review 5000 of 25000

Review 6000 of 25000

Review 7000 of 25000

Review 8000 of 25000

Review 9000 of 25000

Review 10000 of 25000

Review 11000 of 25000

Review 12000 of 25000

Review 13000 of 25000

Review 14000 of 25000

Review 15000 of 25000

Review 16000 of 25000

Review 17000 of 25000

Review 18000 of 25000

Review 19000 of 25000

Review 20000 of 25000

Review 21000 of 25000

Review 22000 of 25000

Review 23000 of 25000

Review 24000 of 25000

Review 25000 of 25000
```

Part2 Word Vectors

读取数据，由于Word2Vec可以从未标记的数据中学习，所以现在可以使用这些额外的50,000条评论。

In [37]:

```
# Read data from files
train = pd.read_csv( "./data/labeledTrainData.tsv", header=0,
 delimiter="\t", quoting=3 )
test = pd.read_csv( "./data/testData.tsv", header=0, delimiter="\t", quoting=3 )
unlabeled_train = pd.read_csv( "./data/unlabeledTrainData.tsv", header=0,
 delimiter="\t", quoting=3 )

# Verify the number of reviews that were read (100,000 in total)
print("Read %d labeled train reviews, %d labeled test reviews, " \
 "and %d unlabeled reviews\n" % (train["review"].size,
 test["review"].size, unlabeled_train["review"].size ))
```

```
Read 25000 labeled train reviews, 25000 labeled test reviews, and 50000 unlabeled reviews
```

重写review_to_wordlist函数，使得去掉停用词成为可选，因为Word2Vec更多地依赖与上下文的语义，而且Word2Vec期望的输入是一个list类型的评论，所以我们不需要将单个的单词合成评论。

In [38]:

```
def review_to_wordlist( review, remove_stopwords=False ):
    # Function to convert a document to a sequence of words,
    # optionally removing stop words.  Returns a list of words.
    #
    # 1. Remove HTML
    review_text = BeautifulSoup(review).get_text()
    #
    # 2. Remove non-letters
    review_text = re.sub("[^a-zA-Z]"," ", review_text)
    #
    # 3. Convert words to lower case and split them
    words = review_text.lower().split()
    #
    # 4. Optionally remove stop words (false by default)
    if remove_stopwords:
        stops = set(stopwords.words("english"))
        words = [w for w in words if not w in stops]
    #
    # 5. Return a list of words
    return(words)
```

将英文的一个段落划分为单个的句子也是有难度的，在这里，我们使用NLTK的punkt来划分段落。

In [41]:

```
# Load the punkt tokenizer
tokenizer = nltk.data.load('tokenizers/punkt/english.pickle')

# Define a function to split a review into parsed sentences
def review_to_sentences( review, tokenizer, remove_stopwords=False ):
    # Function to split a review into parsed sentences. Returns a
    # list of sentences, where each sentence is a list of words
    #
    # 1. Use the NLTK tokenizer to split the paragraph into sentences
    raw_sentences = tokenizer.tokenize(review.strip())
    #
    # 2. Loop over each sentence
    sentences = []
    for raw_sentence in raw_sentences:
        # If a sentence is empty, skip it
        if len(raw_sentence) > 0:
            # Otherwise, call review_to_wordlist to get a list of words
            sentences.append( review_to_wordlist( raw_sentence, \
              remove_stopwords ))
    #
    # Return the list of sentences (each sentence is a list of words,
    # so this returns a list of lists
    return sentences
```

将数据进行切分，将每个段落划分为单个的句子。

In [42]:

```
sentences = []  # Initialize an empty list of sentences

for review in train["review"]:
    sentences += review_to_sentences(review, tokenizer)

for review in unlabeled_train["review"]:
    sentences += review_to_sentences(review, tokenizer)
```

验证划分正确与否，我们输出任意的句子，看看是不是我们所需要的效果。

In [43]:

```
print(sentences[0])
```

```
['with', 'all', 'this', 'stuff', 'going', 'down', 'at', 'the', 'moment', 'with', 'mj', 'i', 've',
'started', 'listening', 'to', 'his', 'music', 'watching', 'the', 'odd', 'documentary', 'here', 'an
d', 'there', 'watched', 'the', 'wiz', 'and', 'watched', 'moonwalker', 'again']
```

选择模型的参数并进行训练。

In [46]:

```
# Import the built-in logging module and configure it so that Word2Vec
# creates nice output messages
import logging
logging.basicConfig(format='%(asctime)s : %(levelname)s : %(message)s',\
    level=logging.INFO)

# Set values for various parameters
num_features = 300    # Word vector dimensionality
min_word_count = 40   # Minimum word count
num_workers = 4       # Number of threads to run in parallel
context = 10          # Context window size

downsampling = 1e-3   # Downsample setting for frequent words

# Initialize and train the model (this will take some time)
from gensim.models import word2vec
print("Training model...")
model = word2vec.Word2Vec(sentences, workers=num_workers, \
            size=num_features, min_count = min_word_count, \
            window = context, sample = downsampling)

# If you don't plan to train the model any further, calling
# init_sims will make the model much more memory-efficient.
model.init_sims(replace=True)

# It can be helpful to create a meaningful model name and
# save the model for later use. You can load it later using Word2Vec.load()
model_name = "300features_40minwords_10context"
model.save(model_name)
```

```
2019-04-18 00:41:37,475 : INFO : collecting all words and their counts
2019-04-18 00:41:37,476 : INFO : PROGRESS: at sentence #0, processed 0 words, keeping 0 word types
2019-04-18 00:41:37,532 : INFO : PROGRESS: at sentence #10000, processed 225803 words, keeping 177
76 word types
2019-04-18 00:41:37,592 : INFO : PROGRESS: at sentence #20000, processed 451892 words, keeping 249
48 word types
2019-04-18 00:41:37,648 : INFO : PROGRESS: at sentence #30000, processed 671315 words, keeping 300
34 word types
```

```
Training model...
```

```
2019-04-18 00:41:37,704 : INFO : PROGRESS: at sentence #40000, processed 897815 words, keeping 343
48 word types
2019-04-18 00:41:37,756 : INFO : PROGRESS: at sentence #50000, processed 1116963 words, keeping 37
761 word types
2019-04-18 00:41:37,812 : INFO : PROGRESS: at sentence #60000, processed 1338404 words, keeping 40
723 word types
2019-04-18 00:41:37,871 : INFO : PROGRESS: at sentence #70000, processed 1561580 words, keeping 43
333 word types
2019-04-18 00:41:37,927 : INFO : PROGRESS: at sentence #80000, processed 1780887 words, keeping 45
714 word types
2019-04-18 00:41:37,987 : INFO : PROGRESS: at sentence #90000, processed 2004996 words, keeping 48
135 word types
2019-04-18 00:41:38,046 : INFO : PROGRESS: at sentence #100000, processed 2226966 words, keeping 5
0207 word types
2019-04-18 00:41:38,108 : INFO : PROGRESS: at sentence #110000, processed 2446580 words, keeping 5
2081 word types
2019-04-18 00:41:38,172 : INFO : PROGRESS: at sentence #120000, processed 2668775 words, keeping 5
4119 word types
2019-04-18 00:41:38,232 : INFO : PROGRESS: at sentence #130000, processed 2894303 words, keeping 5
5847 word types
2019-04-18 00:41:38,286 : INFO : PROGRESS: at sentence #140000, processed 3107005 words, keeping 5
7346 word types
2019-04-18 00:41:38,351 : INFO : PROGRESS: at sentence #150000, processed 3332627 words, keeping 5
```

```
9055 word types
2019-04-18 00:41:38,414 : INFO : PROGRESS: at sentence #160000, processed 3555315 words, keeping 6
0617 word types
2019-04-18 00:41:38,476 : INFO : PROGRESS: at sentence #170000, processed 3778655 words, keeping 6
2077 word types
2019-04-18 00:41:38,542 : INFO : PROGRESS: at sentence #180000, processed 3999236 words, keeping 6
3496 word types
2019-04-18 00:41:38,600 : INFO : PROGRESS: at sentence #190000, processed 4224449 words, keeping 6
4794 word types
2019-04-18 00:41:38,655 : INFO : PROGRESS: at sentence #200000, processed 4448603 words, keeping 6
6087 word types
2019-04-18 00:41:38,718 : INFO : PROGRESS: at sentence #210000, processed 4669967 words, keeping 6
7390 word types
2019-04-18 00:41:38,774 : INFO : PROGRESS: at sentence #220000, processed 4894968 words, keeping 6
8697 word types
2019-04-18 00:41:38,834 : INFO : PROGRESS: at sentence #230000, processed 5117545 words, keeping 6
9958 word types
2019-04-18 00:41:38,903 : INFO : PROGRESS: at sentence #240000, processed 5345050 words, keeping 7
1167 word types
2019-04-18 00:41:38,967 : INFO : PROGRESS: at sentence #250000, processed 5559165 words, keeping 7
2351 word types
2019-04-18 00:41:39,023 : INFO : PROGRESS: at sentence #260000, processed 5779146 words, keeping 7
3478 word types
2019-04-18 00:41:39,081 : INFO : PROGRESS: at sentence #270000, processed 6000435 words, keeping 7
4767 word types
2019-04-18 00:41:39,133 : INFO : PROGRESS: at sentence #280000, processed 6226314 words, keeping 7
6369 word types
2019-04-18 00:41:39,195 : INFO : PROGRESS: at sentence #290000, processed 6449474 words, keeping 7
7839 word types
2019-04-18 00:41:39,253 : INFO : PROGRESS: at sentence #300000, processed 6674077 words, keeping 7
9171 word types
2019-04-18 00:41:39,309 : INFO : PROGRESS: at sentence #310000, processed 6899391 words, keeping 8
0480 word types
2019-04-18 00:41:39,366 : INFO : PROGRESS: at sentence #320000, processed 7124278 words, keeping 8
1808 word types
2019-04-18 00:41:39,419 : INFO : PROGRESS: at sentence #330000, processed 7346021 words, keeping 8
3030 word types
2019-04-18 00:41:39,478 : INFO : PROGRESS: at sentence #340000, processed 7575533 words, keeping 8
4280 word types
2019-04-18 00:41:39,534 : INFO : PROGRESS: at sentence #350000, processed 7798803 words, keeping 8
5425 word types
2019-04-18 00:41:39,592 : INFO : PROGRESS: at sentence #360000, processed 8019427 words, keeping 8
6596 word types
2019-04-18 00:41:39,657 : INFO : PROGRESS: at sentence #370000, processed 8246619 words, keeping 8
7708 word types
2019-04-18 00:41:39,711 : INFO : PROGRESS: at sentence #380000, processed 8471766 words, keeping 8
8878 word types
2019-04-18 00:41:39,773 : INFO : PROGRESS: at sentence #390000, processed 8701497 words, keeping 8
9907 word types
2019-04-18 00:41:39,831 : INFO : PROGRESS: at sentence #400000, processed 8924446 words, keeping 9
0916 word types
2019-04-18 00:41:39,890 : INFO : PROGRESS: at sentence #410000, processed 9145796 words, keeping 9
1880 word types
2019-04-18 00:41:39,945 : INFO : PROGRESS: at sentence #420000, processed 9366876 words, keeping 9
2912 word types
2019-04-18 00:41:40,005 : INFO : PROGRESS: at sentence #430000, processed 9594413 words, keeping 9
3932 word types
2019-04-18 00:41:40,067 : INFO : PROGRESS: at sentence #440000, processed 9821166 words, keeping 9
4906 word types
2019-04-18 00:41:40,120 : INFO : PROGRESS: at sentence #450000, processed 10044928 words, keeping
96036 word types
2019-04-18 00:41:40,184 : INFO : PROGRESS: at sentence #460000, processed 10277688 words, keeping
97088 word types
2019-04-18 00:41:40,243 : INFO : PROGRESS: at sentence #470000, processed 10505613 words, keeping
97933 word types
2019-04-18 00:41:40,301 : INFO : PROGRESS: at sentence #480000, processed 10725997 words, keeping
98862 word types
2019-04-18 00:41:40,364 : INFO : PROGRESS: at sentence #490000, processed 10952741 words, keeping
99871 word types
2019-04-18 00:41:40,422 : INFO : PROGRESS: at sentence #500000, processed 11174397 words, keeping
100765 word types
2019-04-18 00:41:40,481 : INFO : PROGRESS: at sentence #510000, processed 11399672 words, keeping
101699 word types
2019-04-18 00:41:40,542 : INFO : PROGRESS: at sentence #520000, processed 11623020 words, keeping
102598 word types
2019-04-18 00:41:40,605 : INFO : PROGRESS: at sentence #530000, processed 11847418 words, keeping
103400 word types
2019-04-18 00:41:40,666 : INFO : PROGRESS: at sentence #540000, processed 12072033 words, keeping
```

```
2019-04-18 00:41:40,666 : INFO : PROGRESS: at sentence #540000, processed 12072033 words, keeping
104265 word types
2019-04-18 00:41:40,722 : INFO : PROGRESS: at sentence #550000, processed 12297571 words, keeping
105133 word types
2019-04-18 00:41:40,779 : INFO : PROGRESS: at sentence #560000, processed 12518861 words, keeping
105997 word types
2019-04-18 00:41:40,839 : INFO : PROGRESS: at sentence #570000, processed 12747916 words, keeping
106787 word types
2019-04-18 00:41:40,899 : INFO : PROGRESS: at sentence #580000, processed 12969412 words, keeping
107665 word types
2019-04-18 00:41:40,959 : INFO : PROGRESS: at sentence #590000, processed 13194937 words, keeping
108501 word types
2019-04-18 00:41:41,021 : INFO : PROGRESS: at sentence #600000, processed 13417135 words, keeping
109218 word types
2019-04-18 00:41:41,081 : INFO : PROGRESS: at sentence #610000, processed 13638158 words, keeping
110092 word types
2019-04-18 00:41:41,143 : INFO : PROGRESS: at sentence #620000, processed 13864483 words, keeping
110837 word types
2019-04-18 00:41:41,198 : INFO : PROGRESS: at sentence #630000, processed 14088769 words, keeping
111610 word types
2019-04-18 00:41:41,260 : INFO : PROGRESS: at sentence #640000, processed 14309552 words, keeping
112416 word types
2019-04-18 00:41:41,326 : INFO : PROGRESS: at sentence #650000, processed 14535308 words, keeping
113196 word types
2019-04-18 00:41:41,390 : INFO : PROGRESS: at sentence #660000, processed 14758098 words, keeping
113945 word types
2019-04-18 00:41:41,451 : INFO : PROGRESS: at sentence #670000, processed 14981482 words, keeping
114643 word types
2019-04-18 00:41:41,515 : INFO : PROGRESS: at sentence #680000, processed 15206314 words, keeping
115354 word types
2019-04-18 00:41:41,579 : INFO : PROGRESS: at sentence #690000, processed 15428507 words, keeping
116131 word types
2019-04-18 00:41:41,641 : INFO : PROGRESS: at sentence #700000, processed 15657213 words, keeping
116943 word types
2019-04-18 00:41:41,706 : INFO : PROGRESS: at sentence #710000, processed 15880202 words, keeping
117596 word types
2019-04-18 00:41:41,763 : INFO : PROGRESS: at sentence #720000, processed 16105489 words, keeping
118221 word types
2019-04-18 00:41:41,817 : INFO : PROGRESS: at sentence #730000, processed 16331870 words, keeping
118954 word types
2019-04-18 00:41:41,879 : INFO : PROGRESS: at sentence #740000, processed 16552903 words, keeping
119668 word types
2019-04-18 00:41:41,932 : INFO : PROGRESS: at sentence #750000, processed 16771230 words, keeping
120295 word types
2019-04-18 00:41:41,993 : INFO : PROGRESS: at sentence #760000, processed 16990622 words, keeping
120930 word types
2019-04-18 00:41:42,055 : INFO : PROGRESS: at sentence #770000, processed 17217759 words, keeping
121703 word types
2019-04-18 00:41:42,122 : INFO : PROGRESS: at sentence #780000, processed 17447905 words, keeping
122402 word types
2019-04-18 00:41:42,186 : INFO : PROGRESS: at sentence #790000, processed 17674981 words, keeping
123066 word types
2019-04-18 00:41:42,221 : INFO : collected 123504 word types from a corpus of 17798082 raw words a
nd 795538 sentences
2019-04-18 00:41:42,222 : INFO : Loading a fresh vocabulary
2019-04-18 00:41:42,314 : INFO : effective_min_count=40 retains 16490 unique words (13% of
original 123504, drops 107014)
2019-04-18 00:41:42,315 : INFO : effective_min_count=40 leaves 17238940 word corpus (96% of origin
al 17798082, drops 559142)
2019-04-18 00:41:42,384 : INFO : deleting the raw counts dictionary of 123504 items
2019-04-18 00:41:42,388 : INFO : sample=0.001 downsamples 48 most-common words
2019-04-18 00:41:42,389 : INFO : downsampling leaves estimated 12749658 word corpus (74.0% of prio
r 17238940)
2019-04-18 00:41:42,458 : INFO : estimated required memory for 16490 words and 300 dimensions: 478
21000 bytes
2019-04-18 00:41:42,459 : INFO : resetting layer weights
2019-04-18 00:41:42,792 : INFO : training model with 4 workers on 16490 vocabulary and 300
features, using sg=0 hs=0 sample=0.001 negative=5 window=10
2019-04-18 00:41:43,808 : INFO : EPOCH 1 - PROGRESS: at 6.04% examples, 767822 words/s, in_qsize 7
, out_qsize 0
2019-04-18 00:41:44,824 : INFO : EPOCH 1 - PROGRESS: at 12.80% examples, 803198 words/s, in_qsize
7, out_qsize 0
2019-04-18 00:41:45,825 : INFO : EPOCH 1 - PROGRESS: at 19.81% examples, 830524 words/s, in_qsize
7, out_qsize 0
2019-04-18 00:41:46,834 : INFO : EPOCH 1 - PROGRESS: at 26.51% examples, 833821 words/s, in_qsize
7, out_qsize 0
2019-04-18 00:41:47,834 : INFO : EPOCH 1 - PROGRESS: at 33.44% examples, 841121 words/s, in_qsize
```

```
7, out_qsize 0
2019-04-18 00:41:48,838 : INFO : EPOCH 1 - PROGRESS: at 40.18% examples, 844536 words/s, in_qsize
7, out_qsize 0
2019-04-18 00:41:49,847 : INFO : EPOCH 1 - PROGRESS: at 46.72% examples, 842388 words/s, in_qsize
7, out_qsize 0
2019-04-18 00:41:50,855 : INFO : EPOCH 1 - PROGRESS: at 53.67% examples, 847090 words/s, in_qsize
7, out_qsize 0
2019-04-18 00:41:51,859 : INFO : EPOCH 1 - PROGRESS: at 59.55% examples, 837593 words/s, in_qsize
7, out_qsize 0
2019-04-18 00:41:52,867 : INFO : EPOCH 1 - PROGRESS: at 65.95% examples, 834708 words/s, in_qsize
7, out_qsize 0
2019-04-18 00:41:53,876 : INFO : EPOCH 1 - PROGRESS: at 71.99% examples, 828428 words/s, in_qsize
7, out_qsize 0
2019-04-18 00:41:54,884 : INFO : EPOCH 1 - PROGRESS: at 78.95% examples, 832723 words/s, in_qsize
7, out_qsize 0
2019-04-18 00:41:55,892 : INFO : EPOCH 1 - PROGRESS: at 85.74% examples, 834702 words/s, in_qsize
7, out_qsize 0
2019-04-18 00:41:56,897 : INFO : EPOCH 1 - PROGRESS: at 92.74% examples, 838569 words/s, in_qsize
7, out_qsize 0
2019-04-18 00:41:57,903 : INFO : EPOCH 1 - PROGRESS: at 98.77% examples, 833904 words/s, in_qsize
7, out_qsize 0
2019-04-18 00:41:58,061 : INFO : worker thread finished; awaiting finish of 3 more threads
2019-04-18 00:41:58,069 : INFO : worker thread finished; awaiting finish of 2 more threads
2019-04-18 00:41:58,081 : INFO : worker thread finished; awaiting finish of 1 more threads
2019-04-18 00:41:58,085 : INFO : worker thread finished; awaiting finish of 0 more threads
2019-04-18 00:41:58,087 : INFO : EPOCH - 1 : training on 17798082 raw words (12749232 effective wo
rds) took 15.3s, 834107 effective words/s
2019-04-18 00:41:59,101 : INFO : EPOCH 2 - PROGRESS: at 5.60% examples, 713961 words/s, in_qsize 7
, out_qsize 0
2019-04-18 00:42:00,112 : INFO : EPOCH 2 - PROGRESS: at 11.89% examples, 749652 words/s, in_qsize
7, out_qsize 0
2019-04-18 00:42:01,114 : INFO : EPOCH 2 - PROGRESS: at 18.69% examples, 785165 words/s, in_qsize
7, out_qsize 0
2019-04-18 00:42:02,125 : INFO : EPOCH 2 - PROGRESS: at 25.16% examples, 792339 words/s, in_qsize
8, out_qsize 0
2019-04-18 00:42:03,134 : INFO : EPOCH 2 - PROGRESS: at 30.62% examples, 771027 words/s, in_qsize
7, out_qsize 0
2019-04-18 00:42:04,146 : INFO : EPOCH 2 - PROGRESS: at 37.40% examples, 783905 words/s, in_qsize
7, out_qsize 0
2019-04-18 00:42:05,146 : INFO : EPOCH 2 - PROGRESS: at 43.92% examples, 791312 words/s, in_qsize
7, out_qsize 0
2019-04-18 00:42:06,150 : INFO : EPOCH 2 - PROGRESS: at 50.68% examples, 800209 words/s, in_qsize
7, out_qsize 0
2019-04-18 00:42:07,154 : INFO : EPOCH 2 - PROGRESS: at 57.15% examples, 803004 words/s, in_qsize
7, out_qsize 0
2019-04-18 00:42:08,156 : INFO : EPOCH 2 - PROGRESS: at 63.89% examples, 809041 words/s, in_qsize
7, out_qsize 0
2019-04-18 00:42:09,171 : INFO : EPOCH 2 - PROGRESS: at 70.77% examples, 814383 words/s, in_qsize
8, out_qsize 0
2019-04-18 00:42:10,178 : INFO : EPOCH 2 - PROGRESS: at 77.26% examples, 815136 words/s, in_qsize
7, out_qsize 0
2019-04-18 00:42:11,190 : INFO : EPOCH 2 - PROGRESS: at 82.67% examples, 804595 words/s, in_qsize
7, out_qsize 0
2019-04-18 00:42:12,197 : INFO : EPOCH 2 - PROGRESS: at 88.92% examples, 804017 words/s, in_qsize
8, out_qsize 1
2019-04-18 00:42:13,201 : INFO : EPOCH 2 - PROGRESS: at 95.65% examples, 806859 words/s, in_qsize
7, out_qsize 0
2019-04-18 00:42:13,894 : INFO : worker thread finished; awaiting finish of 3 more threads
2019-04-18 00:42:13,898 : INFO : worker thread finished; awaiting finish of 2 more threads
2019-04-18 00:42:13,904 : INFO : worker thread finished; awaiting finish of 1 more threads
2019-04-18 00:42:13,923 : INFO : worker thread finished; awaiting finish of 0 more threads
2019-04-18 00:42:13,925 : INFO : EPOCH - 2 : training on 17798082 raw words (12750739 effective wo
rds) took 15.8s, 805649 effective words/s
2019-04-18 00:42:14,939 : INFO : EPOCH 3 - PROGRESS: at 6.35% examples, 804157 words/s, in_qsize 7
, out_qsize 0
2019-04-18 00:42:15,941 : INFO : EPOCH 3 - PROGRESS: at 12.98% examples, 819343 words/s, in_qsize
7, out_qsize 0
2019-04-18 00:42:16,956 : INFO : EPOCH 3 - PROGRESS: at 19.88% examples, 832963 words/s, in_qsize
7, out_qsize 0
2019-04-18 00:42:17,969 : INFO : EPOCH 3 - PROGRESS: at 26.57% examples, 834662 words/s, in_qsize
7, out_qsize 0
2019-04-18 00:42:18,973 : INFO : EPOCH 3 - PROGRESS: at 33.66% examples, 845562 words/s, in_qsize
7, out_qsize 0
2019-04-18 00:42:19,980 : INFO : EPOCH 3 - PROGRESS: at 40.63% examples, 852483 words/s, in_qsize
7, out_qsize 0
2019-04-18 00:42:20,989 : INFO : EPOCH 3 - PROGRESS: at 47.79% examples, 860287 words/s, in_qsize
7, out_qsize 0
```

```
2019-04-18 00:42:22,002 : INFO : EPOCH 3 - PROGRESS: at 54.49% examples, 858755 words/s, in_qsize
8, out_qsize 0
2019-04-18 00:42:23,006 : INFO : EPOCH 3 - PROGRESS: at 61.02% examples, 856655 words/s, in_qsize
7, out_qsize 0
2019-04-18 00:42:24,010 : INFO : EPOCH 3 - PROGRESS: at 67.86% examples, 857920 words/s, in_qsize
7, out_qsize 0
2019-04-18 00:42:25,010 : INFO : EPOCH 3 - PROGRESS: at 74.34% examples, 855376 words/s, in_qsize
7, out_qsize 0
2019-04-18 00:42:26,013 : INFO : EPOCH 3 - PROGRESS: at 80.26% examples, 846518 words/s, in_qsize
7, out_qsize 0
2019-04-18 00:42:27,023 : INFO : EPOCH 3 - PROGRESS: at 86.25% examples, 839761 words/s, in_qsize
7, out_qsize 0
2019-04-18 00:42:28,023 : INFO : EPOCH 3 - PROGRESS: at 91.82% examples, 830849 words/s, in_qsize
7, out_qsize 0
2019-04-18 00:42:29,026 : INFO : EPOCH 3 - PROGRESS: at 97.62% examples, 824411 words/s, in_qsize
7, out_qsize 0
2019-04-18 00:42:29,357 : INFO : worker thread finished; awaiting finish of 3 more threads
2019-04-18 00:42:29,374 : INFO : worker thread finished; awaiting finish of 2 more threads
2019-04-18 00:42:29,376 : INFO : worker thread finished; awaiting finish of 1 more threads
2019-04-18 00:42:29,380 : INFO : worker thread finished; awaiting finish of 0 more threads
2019-04-18 00:42:29,382 : INFO : EPOCH - 3 : training on 17798082 raw words (12749234 effective wo
rds) took 15.4s, 825329 effective words/s
2019-04-18 00:42:30,394 : INFO : EPOCH 4 - PROGRESS: at 6.45% examples, 819409 words/s, in_qsize 8
, out_qsize 0
2019-04-18 00:42:31,395 : INFO : EPOCH 4 - PROGRESS: at 13.43% examples, 849703 words/s, in_qsize
7, out_qsize 0
2019-04-18 00:42:32,395 : INFO : EPOCH 4 - PROGRESS: at 20.56% examples, 866442 words/s, in_qsize
7, out_qsize 0
2019-04-18 00:42:33,409 : INFO : EPOCH 4 - PROGRESS: at 27.52% examples, 868528 words/s, in_qsize
7, out_qsize 0
2019-04-18 00:42:34,416 : INFO : EPOCH 4 - PROGRESS: at 34.60% examples, 871957 words/s, in_qsize
7, out_qsize 0
2019-04-18 00:42:35,417 : INFO : EPOCH 4 - PROGRESS: at 40.63% examples, 855173 words/s, in_qsize
7, out_qsize 0
2019-04-18 00:42:36,424 : INFO : EPOCH 4 - PROGRESS: at 46.78% examples, 844642 words/s, in_qsize
6, out_qsize 1
2019-04-18 00:42:37,434 : INFO : EPOCH 4 - PROGRESS: at 53.56% examples, 846115 words/s, in_qsize
7, out_qsize 0
2019-04-18 00:42:38,437 : INFO : EPOCH 4 - PROGRESS: at 60.47% examples, 851008 words/s, in_qsize
7, out_qsize 0
2019-04-18 00:42:39,442 : INFO : EPOCH 4 - PROGRESS: at 67.34% examples, 853585 words/s, in_qsize
7, out_qsize 0
2019-04-18 00:42:40,444 : INFO : EPOCH 4 - PROGRESS: at 73.74% examples, 850030 words/s, in_qsize
7, out_qsize 0
2019-04-18 00:42:41,448 : INFO : EPOCH 4 - PROGRESS: at 80.53% examples, 850932 words/s, in_qsize
7, out_qsize 0
2019-04-18 00:42:42,458 : INFO : EPOCH 4 - PROGRESS: at 87.42% examples, 852516 words/s, in_qsize
7, out_qsize 0
2019-04-18 00:42:43,467 : INFO : EPOCH 4 - PROGRESS: at 94.43% examples, 854874 words/s, in_qsize
7, out_qsize 0
2019-04-18 00:42:44,286 : INFO : worker thread finished; awaiting finish of 3 more threads
2019-04-18 00:42:44,289 : INFO : worker thread finished; awaiting finish of 2 more threads
2019-04-18 00:42:44,311 : INFO : worker thread finished; awaiting finish of 1 more threads
2019-04-18 00:42:44,314 : INFO : worker thread finished; awaiting finish of 0 more threads
2019-04-18 00:42:44,315 : INFO : EPOCH - 4 : training on 17798082 raw words (12748728 effective wo
rds) took 14.9s, 854217 effective words/s
2019-04-18 00:42:45,332 : INFO : EPOCH 5 - PROGRESS: at 6.74% examples, 850476 words/s, in_qsize 7
, out_qsize 0
2019-04-18 00:42:46,342 : INFO : EPOCH 5 - PROGRESS: at 13.08% examples, 822070 words/s, in_qsize
7, out_qsize 0
2019-04-18 00:42:47,346 : INFO : EPOCH 5 - PROGRESS: at 19.76% examples, 828256 words/s, in_qsize
7, out_qsize 0
2019-04-18 00:42:48,347 : INFO : EPOCH 5 - PROGRESS: at 26.75% examples, 842544 words/s, in_qsize
7, out_qsize 0
2019-04-18 00:42:49,349 : INFO : EPOCH 5 - PROGRESS: at 33.83% examples, 852023 words/s, in_qsize
7, out_qsize 0
2019-04-18 00:42:50,350 : INFO : EPOCH 5 - PROGRESS: at 40.74% examples, 857743 words/s, in_qsize
7, out_qsize 0
2019-04-18 00:42:51,351 : INFO : EPOCH 5 - PROGRESS: at 47.79% examples, 863788 words/s, in_qsize
7, out_qsize 0
2019-04-18 00:42:52,365 : INFO : EPOCH 5 - PROGRESS: at 54.65% examples, 864304 words/s, in_qsize
6, out_qsize 1
2019-04-18 00:42:53,366 : INFO : EPOCH 5 - PROGRESS: at 61.70% examples, 869078 words/s, in_qsize
7, out_qsize 0
2019-04-18 00:42:54,375 : INFO : EPOCH 5 - PROGRESS: at 68.93% examples, 873615 words/s, in_qsize
7, out_qsize 0
2019-04-18 00:42:55,380 : INFO : EPOCH 5 - PROGRESS: at 75.65% examples, 871866 words/s, in_qsize
```

```
7, out_qsize 0
2019-04-18 00:42:56,390 : INFO : EPOCH 5 - PROGRESS: at 82.27% examples, 868789 words/s, in_qsize
7, out_qsize 0
2019-04-18 00:42:57,397 : INFO : EPOCH 5 - PROGRESS: at 89.21% examples, 869682 words/s, in_qsize
6, out_qsize 1
2019-04-18 00:42:58,405 : INFO : EPOCH 5 - PROGRESS: at 95.53% examples, 864308 words/s, in_qsize
7, out_qsize 0
2019-04-18 00:42:59,053 : INFO : worker thread finished; awaiting finish of 3 more threads
2019-04-18 00:42:59,061 : INFO : worker thread finished; awaiting finish of 2 more threads
2019-04-18 00:42:59,065 : INFO : worker thread finished; awaiting finish of 1 more threads
2019-04-18 00:42:59,078 : INFO : worker thread finished; awaiting finish of 0 more threads
2019-04-18 00:42:59,080 : INFO : EPOCH - 5 : training on 17798082 raw words (12750119 effective wo
rds) took 14.8s, 864066 effective words/s
2019-04-18 00:42:59,081 : INFO : training on a 88990410 raw words (63748052 effective words) took
76.3s, 835619 effective words/s
2019-04-18 00:42:59,083 : INFO : precomputing L2-norms of word weight vectors
2019-04-18 00:42:59,417 : INFO : saving Word2Vec object under 300features_40minwords_10context, se
parately None
2019-04-18 00:42:59,418 : INFO : not storing attribute vectors_norm
2019-04-18 00:42:59,419 : INFO : not storing attribute cum_table
2019-04-18 00:42:59,421 : WARNING : this function is deprecated, use smart_open.open instead
2019-04-18 00:43:00,210 : INFO : saved 300features_40minwords_10context
```

查看我们的模型，doesnt_match给出集合中与其他最不相似的单词，我们可以观察到结果有一部分表现得也不好。

In [44]:

```
model.doesnt_match("man woman child kitchen".split())
```

Out[44]:

```
'kitchen'
```

In [45]:

```
model.doesnt_match("france england germany berlin".split())
```

Out[45]:

```
'berlin'
```

In [46]:

```
model.doesnt_match("paris berlin london austria".split())
```

Out[46]:

```
'london'
```

most_similar可以查看模型的词簇

In [47]:

```
model.most_similar("man")
```

Out[47]:

```
[('woman', 0.6206297874450684),
 ('lad', 0.612740159034729),
 ('lady', 0.5837982892990112),
 ('monk', 0.5688730478286743),
 ('guy', 0.5299125909805298),
 ('men', 0.5279783010482788),
 ('soldier', 0.5218664407730103),
 ('farmer', 0.5183367729187012),
 ('chap', 0.508196234703064),
 ('person', 0.5062509775161743)]
```

In [48]:

```
model.most_similar("queen")
```

Out[48]:

```
[('princess', 0.6679354906082153),
 ('bride', 0.6081489324569702),
 ('maid', 0.5961588621139526),
 ('stepmother', 0.59424889087677),
 ('dame', 0.5883375406265259),
 ('maria', 0.5822771787643433),
 ('duchess', 0.5811522006988525),
 ('mistress', 0.5799473524093628),
 ('victoria', 0.5774447917938232),
 ('latifah', 0.5747784972190857)]
```

Part3 More Fun with Word Vectors

load model，加载我们保存好的模型，不用每次启动都训练一次模型。

In [49]:

```
model = Word2Vec.load("300features_40minwords_10context")
```

模型得到的是一个单词表，每个单词用一个size为300的features的向量表示。

In [50]:

```
type(model.wv.syn0)
model["flower"].shape
```

Out[50]:

```
(300,)
```

由于每个单词都是300维空间中的向量，我们可以使用向量运算来组合每个评论中的单词。 我们尝试的一种方法是在给定的评论中简单地平均单词向量（为此，我们删除了停止单词，这只会增加噪音）。

In [60]:

```
def makeFeatureVec(words, model, num_features):
    # Function to average all of the word vectors in a given
    # paragraph
    #
    # Pre-initialize an empty numpy array (for speed)
    featureVec = np.zeros((num_features,),dtype="float32")
    #
    nwords = 0.
    #
    # Index2word is a list that contains the names of the words in
    # the model's vocabulary. Convert it to a set, for speed
    index2word_set = set(model.wv.index2word)
    #
    # Loop over each word in the review and, if it is in the model's
    # vocaublary, add its feature vector to the total
    for word in words:
        if word in index2word_set:
            nwords = nwords + 1.
            featureVec = np.add(featureVec,model[word])
    #
    # Divide the result by the number of words to get the average
    featureVec = np.divide(featureVec,nwords)
    return featureVec


def getAvgFeatureVecs(reviews, model, num_features):
    # Given a set of reviews (each one a list of words), calculate
    # the average feature vector for each one and return a 2D numpy array
    #
    # Initialize a counter
    counter = 0
```

```
    #
    # Preallocate a 2D numpy array, for speed
    reviewFeatureVecs = np.zeros((len(reviews),num_features),dtype="float32")
    #
    # Loop through the reviews
    for review in reviews:
       if counter%1000== 0:
           print("Review %d of %d" % (counter, len(reviews)))
       #
       # Call the function (defined above) that makes average feature vectors
       reviewFeatureVecs[counter] = makeFeatureVec(review, model, num_features)
       #
       # Increment the counter
       counter = counter + 1
    return reviewFeatureVecs
```

In [61]:

```
# Calculate average feature vectors for training and testing sets,
# using the functions we defined above. Notice that we now use stop word
# removal.

clean_train_reviews = []
for review in train["review"]:
    clean_train_reviews.append( review_to_wordlist( review, \
        remove_stopwords=True ))

trainDataVecs = getAvgFeatureVecs( clean_train_reviews, model, len(model['flower']) )

clean_test_reviews = []
for review in test["review"]:
    clean_test_reviews.append( review_to_wordlist( review, \
        remove_stopwords=True ))

testDataVecs = getAvgFeatureVecs( clean_test_reviews, model, len(model['flower']) )
```

```
Review 0 of 25000
Review 1000 of 25000
Review 2000 of 25000
Review 3000 of 25000
Review 4000 of 25000
Review 5000 of 25000
Review 6000 of 25000
Review 7000 of 25000
Review 8000 of 25000
Review 9000 of 25000
Review 10000 of 25000
Review 11000 of 25000
Review 12000 of 25000
Review 13000 of 25000
Review 14000 of 25000
Review 15000 of 25000
Review 16000 of 25000
Review 17000 of 25000
Review 18000 of 25000
Review 19000 of 25000
Review 20000 of 25000
Review 21000 of 25000
Review 22000 of 25000
Review 23000 of 25000
Review 24000 of 25000
Review 0 of 25000
Review 1000 of 25000
Review 2000 of 25000
Review 3000 of 25000
Review 4000 of 25000
Review 5000 of 25000
Review 6000 of 25000
Review 7000 of 25000
Review 8000 of 25000
Review 9000 of 25000
Review 10000 of 25000
Review 11000 of 25000
Review 12000 of 25000
Review 13000 of 25000
```

```
Review 14000 of 25000
Review 15000 of 25000
Review 16000 of 25000
Review 17000 of 25000
Review 18000 of 25000
Review 19000 of 25000
Review 20000 of 25000
Review 21000 of 25000
Review 22000 of 25000
Review 23000 of 25000
Review 24000 of 25000
```

然后使用评论的平均向量来训练一个随机森林。最后我们提交答案时发现这种方法要比单词袋的效果要差一点。

In [63]:

```
# Fit a random forest to the training data, using 100 trees
forest = RandomForestClassifier( n_estimators = 100 )

forest = forest.fit( trainDataVecs, train["sentiment"] )

# Test & extract results
result = forest.predict( testDataVecs )

# Write the test results
output = pd.DataFrame( data={"id":test["id"], "sentiment":result} )
output.to_csv( "Word2Vec_AverageVectors.csv", index=False, quoting=3 )
```

聚簇：Word2Vec创建语义相关的单词集群，因此另一种可能的方法是利用集群内单词的相似性。K-Means是常见的聚簇算法。实验证明每个聚簇要5个单词的效果是比较好的。下面我们将出现的词汇表变成聚簇。

In [68]:

```
start = time.time() # Start time

# Set "k" (num_clusters) to be 1/5th of the vocabulary size, or an
# average of 5 words per cluster
word_vectors = model.wv.syn0
num_clusters = (int)(word_vectors.shape[0] / 5)

# Initalize a k-means object and use it to extract centroids
kmeans_clustering = KMeans( n_clusters = num_clusters )
idx = kmeans_clustering.fit_predict( word_vectors )

# Get the end time and print how long the process took
end = time.time()
elapsed = end - start
print("Time taken for K Means clustering: ", elapsed, "seconds.")
```

```
Time taken for K Means clustering:  648.8692169189453 seconds.
```

同样，像part1一样，我们将每一段评论转换成一个向量，但与part1不同的是我们的计数不是以单词为基础，而是以聚簇为基础的。为了方便，我们将单词用词典的方式表示，与聚簇的id对应，最后每段评论可以表示为一个size为聚簇的个数的向量。

In [72]:

```
# Create a Word / Index dictionary, mapping each vocabulary word to
# a cluster number

word_centroid_map = dict(zip( model.wv.index2word, idx ))
```

In [73]:

```
def create_bag_of_centroids( wordlist, word_centroid_map ):
    #
    # The number of clusters is equal to the highest cluster index
    # in the word / centroid map
    num_centroids = max( word_centroid_map.values() ) + 1
    #
    # Pre-allocate the bag of centroids vector (for speed)
```

```
    bag_of_centroids = np.zeros( num_centroids, dtype="float32" )
    #
    # Loop over the words in the review. If the word is in the vocabulary,
    # find which cluster it belongs to, and increment that cluster count
    # by one
    for word in wordlist:
        if word in word_centroid_map:
            index = word_centroid_map[word]
            bag_of_centroids[index] += 1
    #
    # Return the "bag of centroids"
    return bag_of_centroids
```

训练我们的随机森林模型然后预测测试集的结果。最后结果Bag of Words的预测的效果差不多。

In [74]:

```
# Pre-allocate an array for the training set bags of centroids (for speed)
train_centroids = np.zeros( (train["review"].size, num_clusters), \
    dtype="float32" )

# Transform the training set reviews into bags of centroids
counter = 0
for review in clean_train_reviews:
    train_centroids[counter] = create_bag_of_centroids( review, \
        word_centroid_map )
    counter += 1

# Repeat for test reviews
test_centroids = np.zeros(( test["review"].size, num_clusters), \
    dtype="float32" )

counter = 0
for review in clean_test_reviews:
    test_centroids[counter] = create_bag_of_centroids( review, \
        word_centroid_map )
    counter += 1
```

In [76]:

```
# Fit a random forest and extract predictions
forest = RandomForestClassifier(n_estimators = 100)

# Fitting the forest may take a few minutes
forest = forest.fit(train_centroids,train["sentiment"])
result = forest.predict(test_centroids)

# Write the test results
output = pd.DataFrame(data={"id":test["id"], "sentiment":result})
output.to_csv( "BagOfCentroids.csv", index=False, quoting=3 )
```

Part4 Comparing

在教程中，平均向量和使用中心线会丢失单词的顺序，使其非常类似于单词包的概念。由于性能相似(在标准误差范围内)，所以这三种方法实际上是等价的。