

In [1]:

```

%matplotlib inline
import warnings
warnings.filterwarnings("ignore")

import sqlite3
import pandas as pd
import numpy as np
import nltk
import string
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.feature_extraction.text import TfidfTransformer
from sklearn.feature_extraction.text import TfidfVectorizer

from sklearn.feature_extraction.text import CountVectorizer
from sklearn.metrics import confusion_matrix
from sklearn import metrics
from sklearn.metrics import roc_curve, auc
from nltk.stem.porter import PorterStemmer

import re
# Tutorial about Python regular expressions: https://pymotw.com/2/re/
import string
from nltk.corpus import stopwords
from nltk.stem import PorterStemmer
from nltk.stem.wordnet import WordNetLemmatizer

from gensim.models import Word2Vec
from gensim.models import KeyedVectors
import pickle

from tqdm import tqdm
import os

```

In [5]:

```

from google.colab import drive
drive.mount('/content/gdrive')

```

Go to this URL in a browser: https://accounts.google.com/o/oauth2/auth?client_id=947318989803-6bn6qk8qdgf4n4g3pfee6491hc0brc4i.apps.googleusercontent.com&redirect_uri=urn%3Aietf%3Awg%3Aoauth%3A2.0%3Aaob&scope=email%20https%3A%2F%2Fwww.googleapis.com%2Fauth%2Fdocs.test%20https%3A%2F%2Fwww.googleapis.com%2Fauth%2Fdrive.photos.readonly%20https%3A%2F%2Fwww.googleapis.com%2Fauth%2Fpeopleapi.readonly&response_type=code (https://accounts.google.com/o/oauth2/auth?client_id=947318989803-6bn6qk8qdgf4n4g3pfee6491hc0brc4i.apps.googleusercontent.com&redirect_uri=urn%3Aietf%3Awg%3Aoauth%3A2.0%3Aaob&scope=email%20https%3A%2F%2Fwww.googleapis.com%2Fauth%2Fdocs.test%20https%3A%2F%2Fwww.googleapis.com%2Fauth%2Fdrive%20https%3A%2F%2Fwww.googleapis.com%2Fauth%2Fdrive.photos.readonly%20https%3A%2F%2Fwww.googleapis.com%2Fauth%2Fpeopleapi.readonly&response_type=code)

Enter your authorization code:

.....

Mounted at /content/gdrive

In [3]:

```

#connecting to sqlite db
# con = sqlite3.connect('/content/gdrive/My Drive/Colab Notebooks/Assignment 4/database.sql
con = sqlite3.connect('database.sqlite')

# filtering only positive and negative reviews i.e.
# not taking into consideration those reviews with Score=3
# SELECT * FROM Reviews WHERE Score != 3 LIMIT 500000, will give top 500000 data points
# you can change the number to any other number based on your computing power

# filtered_data = pd.read_sql_query(""" SELECT * FROM Reviews WHERE Score != 3 LIMIT 500000
# for tsne assignment you can take 5k data points

filtered_data = pd.read_sql_query(""" SELECT * FROM Reviews WHERE Score != 3""", con)

# Give reviews with Score>3 a positive rating(1), and reviews with a score<3 a negative rating(0)
def partition(x):
    if x < 3:
        return 0
    return 1

#changing reviews with score less than 3 to be positive and vice-versa
actualScore = filtered_data['Score']
positiveNegative = actualScore.map(partition)
filtered_data['Score'] = positiveNegative
print("Number of data points in our data", filtered_data.shape)
filtered_data.head(3)

```

Number of data points in our data (525814, 10)

Out[3]:

	Id	ProductId	UserId	ProfileName	HelpfulnessNumerator	HelpfulnessDenominator
0	1	B001E4KFG0	A3SGXH7AUHU8GW	delmartian	1	1
1	2	B00813GRG4	A1D87F6ZCVE5NK	dll pa	0	0
2	3	B000LQOCH0	ABXLMWJIXXAIN	Natalia Corres "Natalia Corres"	1	1

In [4]:

```
display = pd.read_sql_query("""
SELECT UserId, ProductId, ProfileName, Time, Score, Text, COUNT(*)
FROM Reviews
GROUP BY UserId
HAVING COUNT(*)>1
""", con)
```

In [5]:

```
print(display.shape)
display.head()
```

(80668, 7)

Out[5]:

	UserId	ProductId	ProfileName	Time	Score	Text	CO
0	#oc-R115TNMSPFT9I7	B007Y59HVM	Breyton	1331510400	2	Overall its just OK when considering the price...	2
1	#oc-R11D9D7SHXIJB9	B005HG9ET0	Louis E. Emory "hoppy"	1342396800	5	My wife has recurring extreme muscle spasms, u...	3
2	#oc-R11DNU2NBKQ23Z	B007Y59HVM	Kim Cieszykowski	1348531200	1	This coffee is horrible and unfortunately not ...	2
3	#oc-R11O5J5ZVQE25C	B005HG9ET0	Penguin Chick	1346889600	5	This will be the bottle that you grab from the...	3
4	#oc-R12KPBODL2B5ZD	B007OSBE1U	Christopher P. Presta	1348617600	1	I didnt like this coffee. Instead of telling y...	2

In [6]:

```
# Removing duplicate reviews
final=filtered_data.drop_duplicates(subset={"UserId","ProfileName","Time","Text"}, keep='fi
print(final.shape)
```

(364173, 10)

In [7]:

```
(final['Id'].size*1.0)/(filtered_data['Id'].size*1.0)*100
```

Out[7]:

69.25890143662969

In [8]:

```
final=final[final.HelpfulnessNumerator<=final.HelpfulnessDenominator]
```

In [9]:

```
#Before starting the next phase of preprocessing lets see the number of entries left  
print(final.shape)
```

```
#How many positive and negative reviews are present in our dataset?  
final['Score'].value_counts()
```

(364171, 10)

Out[9]:

```
1    307061  
0     57110  
Name: Score, dtype: int64
```

In [17]:

```
final["cleanReview"] = final["Summary"].map(str) + ". " + final["Text"]
```

In [18]:

```
final['cleanReview'].head()
```

Out[18]:

```
0    Good Quality Dog Food. I have bought several o...  
1    Not as Advertised. Product arrived labeled as ...  
2    "Delight" says it all. This is a confection th...  
3    Cough Medicine. If you are looking for the sec...  
4    Great taffy. Great taffy at a great price. Th...  
Name: cleanReview, dtype: object
```

In [38]:

```
final['lengthOfReview'] = final['cleanReview'].str.split().str.len()  
final['lengthOfReview'].head()
```

Out[38]:

```
0    27  
1    21  
2    43  
3    20  
4    15  
Name: lengthOfReview, dtype: int64
```


In [23]:

```
# https://stackoverflow.com/a/47091490/4084039
import re

def decontracted(phrase):
    # specific
    phrase = re.sub(r"won't", "will not", phrase)
    phrase = re.sub(r"can't", "can not", phrase)

    # general
    phrase = re.sub(r"n't", " not", phrase)
    phrase = re.sub(r"'re", " are", phrase)
    phrase = re.sub(r"'s", " is", phrase)
    phrase = re.sub(r"'d", " would", phrase)
    phrase = re.sub(r"'ll", " will", phrase)
    phrase = re.sub(r"'t", " not", phrase)
    phrase = re.sub(r"'ve", " have", phrase)
    phrase = re.sub(r"'m", " am", phrase)
    return phrase
```

In [24]:

```
decat_lst = []
for decat_text in tqdm(text_lst):
    text = decontracted(decat_text)
    decat_lst.append(text)
```

```
100%|████████████████████████████████████████████████████████████████████████████████| 364171/364171 [00:0
5<00:00, 67496.57it/s]
```

In [25]:

```
strip_list = []
for to_strip in tqdm(decat_lst):
    text = re.sub("\S*\d\S*", "", to_strip).strip()
    strip_list.append(text)
```

```
100%|████████████████████████████████████████████████████████████████████████████████| 364171/364171 [00:1
9<00:00, 18494.66it/s]
```

In [26]:

```
spatial_list = []
for to_spatial in tqdm(strip_list):
    text = re.sub('[^A-Za-z0-9]+', ' ', to_spatial)
    spatial_list.append(text)
```

```
100%|████████████████████████████████████████████████████████████████████████████████| 364171/364171 [00:0
9<00:00, 36464.50it/s]
```


In [117]:

```
print(len(final))
final.tail(5)
```

364171

Out[117]:

Userld	ProfileName	HelpfulnessNumerator	HelpfulnessDenominator	Score	Time
054AY	Lettie D. Carter	0	0	1	1299628800
3KI5	R. Sawyer	0	0	0	1331251200
/751Z	pksd "pk_007"	2	2	1	1329782400
NOH	Kathy A. Welch "katwel"	1	1	1	1331596800
/L9UC	srfell17	0	0	1	1338422400

In [118]:

```
final['cleanReview'][0]
```

Out[118]:

```
'good quality dog food bought several vitality canned dog food products found good quality product looks like stew processed meat smells better labrador finicky appreciates product better'
```


In [119]:

```
final['lengthOfReview'][0]
```

Out[119]:

27

In [170]:

```
dir_path = os.getcwd()  
# conn = sqlite3.connect(os.path.join(dir_path, '/content/gdrive/My Drive/Colab Notebooks/A  
conn = sqlite3.connect(os.path.join(dir_path, 'final.sqlite'))  
final.to_sql('Reviews', conn, if_exists='replace', index=False)
```

In [171]:

```
review_3 = pd.read_sql_query(""" SELECT count(*) FROM Reviews""", conn)  
print(review_3)
```

```
   count(*)  
0      364171
```

In [172]:

```
filtered_data = pd.read_sql_query(""" SELECT * FROM Reviews""", conn)
```

In [173]:

```
filtered_data.shape
```

Out[173]:

```
(364171, 12)
```

In [174]:

```
filtered_data["Time"] = pd.to_datetime(filtered_data["Time"], unit = "s")  
filtered_data = filtered_data.sort_values(by = "Time")
```

In [175]:

```
filtered_data.head(5)
```

Out[175]:

	Id	ProductId	UserId	ProfileName	HelpfulnessNumerator
117924	150524	0006641040	ACITT7DI6IDDL	shari zychinski	0
117901	150501	0006641040	AJ46FKXOVC7NR	Nicholas A Mesiano	2
298792	451856	B00004CXX9	AIUWLEQ1ADEG5	Elizabeth Medina	0
169281	230285	B00004RYGX	A344SMIA5JECGM	Vincent P. Ross	1
298791	451855	B00004CXX9	AJH6LUC1UT1ON	The Phantom of the Opera	0



In [176]:

```
print(len(filtered_data))
filtered_data.info()
filtered_data = filtered_data.head(100000)
print(len(filtered_data))
```

```
364171
<class 'pandas.core.frame.DataFrame'>
Int64Index: 364171 entries, 117924 to 107253
Data columns (total 12 columns):
Id                364171 non-null int64
ProductId         364171 non-null object
UserId           364171 non-null object
ProfileName       364171 non-null object
HelpfulnessNumerator  364171 non-null int64
HelpfulnessDenominator 364171 non-null int64
Score            364171 non-null int64
Time             364171 non-null datetime64[ns]
Summary          364171 non-null object
Text            364171 non-null object
cleanReview      364171 non-null object
lengthOfReview   364171 non-null int64
dtypes: datetime64[ns](1), int64(5), object(6)
memory usage: 36.1+ MB
100000
```

In [177]:

```
filtered_data['Score'].value_counts()
```

Out[177]:

```
1    87729
0    12271
Name: Score, dtype: int64
```

In [178]:

```
X = filtered_data["cleanReview"]
print(print("shape of X:", X.head(5)))
y = filtered_data["Score"]
print("shape of y:", y.head(5))
X_len = filtered_data['lengthOfReview']
```

```
shape of X: 117924    every book educational witty little book makes...
117901    whole series great way spend time child rememb...
298792    entertainingl funny beetlejuice well written m...
169281    modern day fairy tale twist rumplestiskin capt...
298791    fantastic beetlejuice excellent funny movie ke...
Name: cleanReview, dtype: object
None
shape of y: 117924    1
117901    1
298792    1
169281    1
298791    1
Name: Score, dtype: int64
```

In [179]:

```
len(filtered_data['lengthOfReview'])
```

Out[179]:

100000

In [180]:

```
X_train = X[0:60000]
Y_train = y[0:60000]
X_val = X[60000:80000]
Y_val = y[60000:80000]
X_test = X[80000:100000]
Y_test = y[80000:100000]
```

In [181]:

```
print(len(X_train), len(X_test), len(X_val))
print(len(Y_train), len(Y_test), len(Y_val))
```

60000 20000 20000
60000 20000 20000

In [182]:

```
print(X_train.shape, Y_train.shape, X_test.shape)
print(X_train.shape, Y_train.shape, X_val.shape)
```

(60000,) (60000,) (20000,)
(60000,) (60000,) (20000,)

Bag of Words

In [185]:

```

from sklearn.feature_extraction.text import CountVectorizer

count_vect = CountVectorizer()
X_train_vect = count_vect.fit_transform(X_train)
X_test_vect = count_vect.transform(X_test)
X_val_vect = count_vect.transform(X_val)
feature_names = count_vect.get_feature_names()
# Bow_dict = {'X_train_vect':X_train_vect, 'X_test_vect': X_test_vect, 'X_val_vect': X_val_
print(X_train_vect.shape)
print(feature_names)

```

uely', 'absoluetly', 'absolut', 'absolutaly', 'absolute', 'absoluteky', 'a
bsolutel', 'absolutelly', 'absolutelty', 'absolutely', 'absolutelydelectab
le', 'absolutelydelicious', 'absolutelyscrumptious', 'absolutley', 'absolu
tly', 'absolutuley', 'absolves', 'absorb', 'absorbable', 'absorbance', 'ab
sorbancy', 'absorbant', 'absorbed', 'absorbency', 'absorbent', 'absorbin
g', 'absorbs', 'absorption', 'absouloutely', 'absoulte', 'absoulutely', 'ab
soulutely', 'absoutely', 'abstain', 'abstained', 'abstaining', 'abstinenc
e', 'abstract', 'abstraction', 'abstracts', 'absolutely', 'absurd', 'absur
dity', 'absurdly', 'abt', 'abudance', 'abuela', 'abuelita', 'abums', 'abun
dance', 'abundant', 'abundantly', 'abundunce', 'abuse', 'abused', 'abuse
s', 'abusing', 'abusive', 'abut', 'abuts', 'abutted', 'abysmal', 'abyssini
an', 'abyssinians', 'ac', 'acacia', 'academy', 'acai', 'acaia', 'acau', 'a
ccasion', 'accedient', 'accelerade', 'acceleradetests', 'accelerate', 'acc
elerated', 'accent', 'accented', 'accenting', 'accents', 'accentuate', 'ac
centuated', 'accepeted', 'accept', 'acceptable', 'acceptably', 'acceptanc
e', 'acceptd', 'accepted', 'accepting', 'accepts', 'accesesible', 'accesib
le', 'access', 'accessable', 'accessed', 'accessibility', 'accessible', 'a
ccessories', 'accessorize', 'accessory', 'accident', 'accidental', 'accide
ntally', 'accidently', 'accidents', 'accughe', 'acclaim', 'acclaimed', 'a
cclamated', 'acclimate', 'acclimated', 'acclimating', 'accolades', 'accoma

In [186]:

```
X_train_vect.shape
```

Out[186]:

(60000, 47535)

In [187]:

```
len(final['lengthOfReview'])
```

Out[187]:

364171

In [191]:

```

from scipy.sparse import hstack
# len_review = final['lengthOfReview'].to_sparse()
concat_data = hstack((X_train_vect,np.array(final['lengthOfReview'])[0:60000])[:,None]))
concat_data_val = hstack((X_val_vect,np.array(final['lengthOfReview'])[60000:80000])[:,None])
concat_data_test = hstack((X_test_vect,np.array(final['lengthOfReview'])[80000:100000])[:,None])

```

In [193]:

```
print(concat_data.shape)
print(concat_data_val.shape)
print(concat_data_test.shape)
```

```
(60000, 47536)
(20000, 47536)
(20000, 47536)
```

In [194]:

```
print(len(feature_names))
```

```
47535
```

In [195]:

```
BoW_dict = {'X_train_vect':concat_data, 'X_test_vect': concat_data_test, 'X_val_vect': concat_data_val}
print(BoW_dict['X_train_vect'].shape)
```

```
(60000, 47536)
```

In [196]:

```
import pickle
with open('BoW.pkl', 'wb') as handle:
    pickle.dump(BoW_dict, handle, protocol=pickle.HIGHEST_PROTOCOL)
```

TF-IDF

In [209]:

```
tf_idf_vect = TfidfVectorizer(ngram_range=(1,2), min_df=10)
train_tf_idf = tf_idf_vect.fit_transform(X_train)
cv_tf_idf = tf_idf_vect.transform(X_val)
test_tf_idf = tf_idf_vect.transform(X_test)

print("the shape of out text TFIDF vectorizer ",train_tf_idf.get_shape())
print("the type of count vectorizer ",type(train_tf_idf))
print("the number of unique words including both unigrams and bigrams ", train_tf_idf.get_feature_names().shape[0])
```

```
the shape of out text TFIDF vectorizer (60000, 35873)
the type of count vectorizer <class 'scipy.sparse.csr.csr_matrix'>
the number of unique words including both unigrams and bigrams 35873
```

In [210]:

```
tfidf_concat_data_train = hstack((train_tf_idf,np.array(final['lengthOfReview'])[0:60000]))[:,:]
tfidf_concat_data_val = hstack((cv_tf_idf,np.array(final['lengthOfReview'])[60000:80000]))[:,:]
tfidf_concat_data_test = hstack((test_tf_idf,np.array(final['lengthOfReview'])[80000:100000]))[:,:]

```

In [211]:

```
tf_idf_dict = {'train_tf_idf': tfidf_concat_data_train, 'cv_tf_idf': tfidf_concat_data_val, 'test_tf_idf': tfidf_concat_data_test}
```

In [212]:

```
import pickle
with open('tf_idf.pkl', 'wb') as handle:
    pickle.dump(tf_idf_dict, handle, protocol=pickle.HIGHEST_PROTOCOL)
```

Naive Bayes on BoW

In [197]:

```
import pickle
# with open(r"/content/gdrive/My Drive/Colab Notebooks/Assignment 4/BoW.pkl", "rb") as input_file:
with open(r"BoW.pkl", "rb") as input_file:
    BoW_dict = pickle.load(input_file)
```

In [215]:

```
#https://stackoverflow.com/questions/26976362/how-to-get-most-informative-features-for-scik
neg_features_labels = []
neg_features_coeff = []
neg_features_feat = []

pos_features_labels = []
pos_features_coeff = []
pos_features_feat = []

def most_informative_feature_for_binary_classification(vectorizer, classifier, n=10):
    class_labels = classifier.classes_
    feature_names = vectorizer.get_feature_names()
    topn_class1 = sorted(zip(classifier.coef_[0], feature_names))[:n]
    topn_class2 = sorted(zip(classifier.coef_[0], feature_names))[-n:]

    for coef, feat in topn_class1:
        neg_features_labels.append(class_labels[0])
        neg_features_coeff.append(coef)
        neg_features_feat.append(feat)

    for coef, feat in reversed(topn_class2):
        pos_features_labels.append(class_labels[1])
        pos_features_coeff.append(coef)
        pos_features_feat.append(feat)

neg_df = pd.DataFrame({'Labels': neg_features_labels, 'Coeff': neg_features_coeff, 'Negat
pos_df = pd.DataFrame({'Labels': pos_features_labels, 'Coeff': pos_features_coeff, 'Negat
print("Top 10 featues for negative class \n", neg_df)
print("Top 10 featues for positive class \n", pos_df)
```

In [199]:

```
from sklearn.naive_bayes import MultinomialNB
from tqdm import tqdm

bow_auc_train = []
bow_auc_cv = []
bow_auc_test = []

for alpha_value in [ 0.00001, 0.0001, 0.001, 0.01, 0.1, 1, 10, 100, 1000, 10000, 100000]:
    nb = MultinomialNB(alpha=alpha_value)
    nb.fit(Bow_dict['X_train_vect'], Y_train)

    train_proba = nb.score(Bow_dict['X_train_vect'], Y_train)
    bow_auc_train.append(train_proba)
    cv_proba = nb.score(Bow_dict['X_val_vect'], Y_val)
    bow_auc_cv.append(cv_proba)

f = most_informative_feature_for_binary_classification(count_vect, nb)
```

Top 10 features for negative class

	Coeff	Labels	Negative features
0	-10.770155	0	aachen
1	-10.770155	0	aafco
2	-10.770155	0	ablaze
3	-10.770155	0	abominable
4	-10.770155	0	abortions
5	-10.770155	0	abbreviating
6	-10.770155	0	abstraction
7	-10.770155	0	abuses
8	-10.770155	0	acau
9	-10.770155	0	accelerate

Top 10 features for positive class

	Coeff	Labels	Negative features
0	-10.392870	1	not
1	-10.541618	1	great
2	-10.563572	1	good
3	-10.582523	1	like
4	-10.590362	1	tea
5	-10.628716	1	taste
6	-10.633982	1	one
7	-10.641234	1	product
8	-10.644660	1	love
9	-10.644730	1	best

In [200]:

```
alpha_list = [ 0.00001, 0.0001, 0.001, 0.01, 0.1, 1, 10, 100, 1000, 10000, 100000]
train_alpha_dict = dict(zip(alpha_list, bow_auc_train))
val_alpha_dict = dict(zip(alpha_list, bow_auc_cv))
print(train_alpha_dict)
print(val_alpha_dict)
```

```
{0.1: 0.94884999999999997, 1: 0.92926666666666669, 100: 0.88581666666666667,
1000: 0.88581666666666667, 10000: 0.88581666666666667, 1e-05: 0.964849999999999
999, 0.0001: 0.96388333333333331, 10: 0.88816666666666666, 100000: 0.8858166
666666667, 0.01: 0.95794999999999997, 0.001: 0.96193333333333331}
{0.1: 0.91659999999999997, 1: 0.90749999999999997, 100: 0.86280000000000001,
1000: 0.86280000000000001, 10000: 0.86280000000000001, 1e-05: 0.905349999999
99999, 0.0001: 0.90834999999999999, 10: 0.86485000000000001, 100000: 0.86280
000000000001, 0.01: 0.91574999999999995, 0.001: 0.91300000000000003}
```

In [201]:

```
bow_best_alpha = max(val_alpha_dict, key=val_alpha_dict.get)
bow_best_alpha
```

Out[201]:

0.1

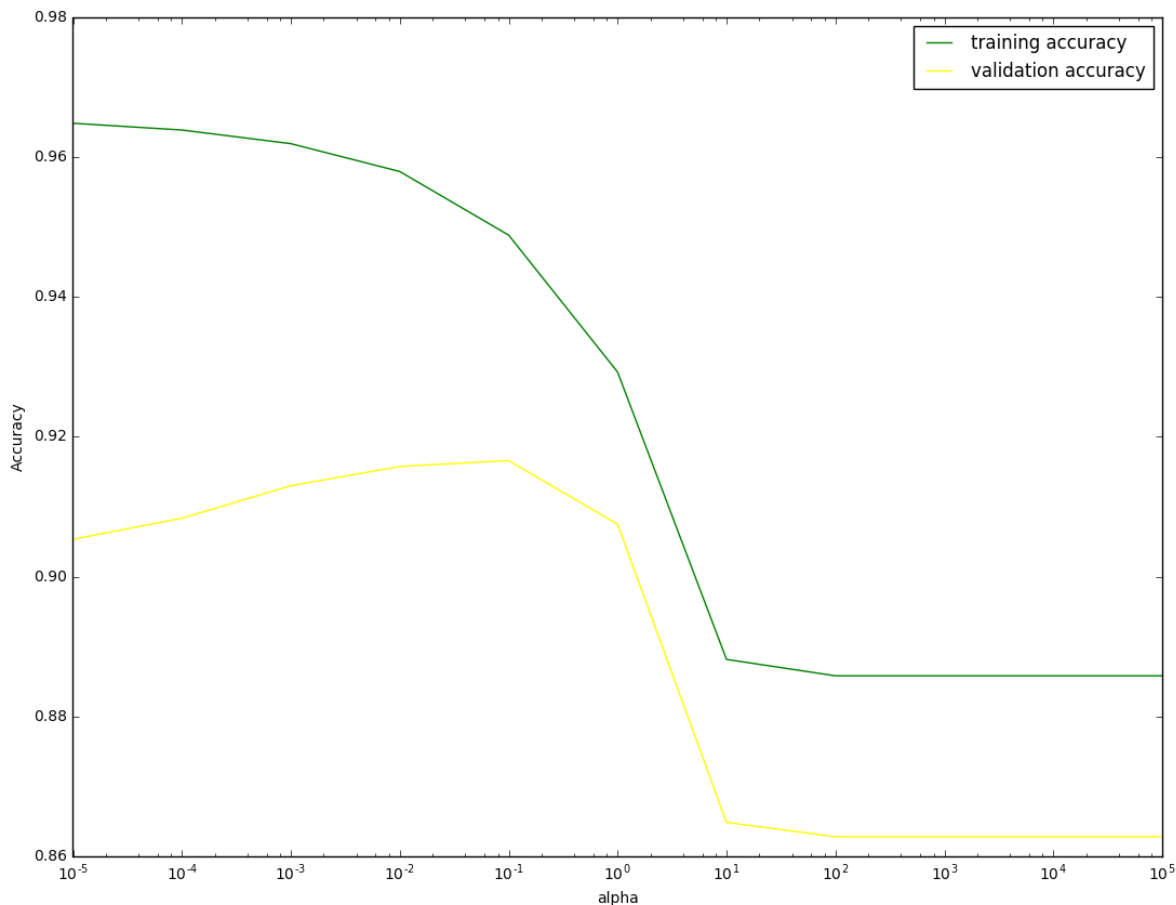
In [202]:

```

import pylab
plt.figure(figsize=(13, 10))
neighbors_settings = [ 0.00001, 0.0001, 0.001, 0.01, 0.1, 1, 10, 100, 1000, 10000, 100000]
plt.plot(neighbors_settings, bow_auc_train, label="training accuracy", color='green')
plt.plot(neighbors_settings, bow_auc_cv, label="validation accuracy", color='yellow')
# plt.plot(neighbors_settings, auc_test, label="test accuracy", color='red')
plt.xlabel('alpha')
plt.ylabel('Accuracy')
plt.legend()
plt.xscale('log')

plt.show()

```



In [203]:

```

nb = MultinomialNB(alpha=bow_best_alpha)
nb.fit(Bow_dict['X_train_vect'], Y_train)
test_predict_bow = nb.predict_proba(Bow_dict['X_test_vect'])
train_predict_bow = nb.predict_proba(Bow_dict['X_train_vect'])

bow_test_conf = nb.predict(Bow_dict['X_test_vect'])
bow_train_conf = nb.predict(Bow_dict['X_train_vect'])

print(type(test_predict_bow))
print(test_predict_bow[:, 1])

<class 'numpy.ndarray'>
[ 9.99996662e-01  4.36918808e-07  1.00000000e+00 ...,  9.94959440e-01
 9.99990491e-01  4.60863281e-02]

```

In [204]:

```
fpr_train, tpr_train, _ = roc_curve(Y_train, train_predict_bow[:, 1])
fpr_test, tpr_test, _ = roc_curve(Y_test, test_predict_bow[:, 1])
bow_test_auc = auc(fpr_test, tpr_test)
bow_train_auc = auc(fpr_train, tpr_train)
print(bow_test_auc)
print(bow_train_auc)
```

0.927312069814

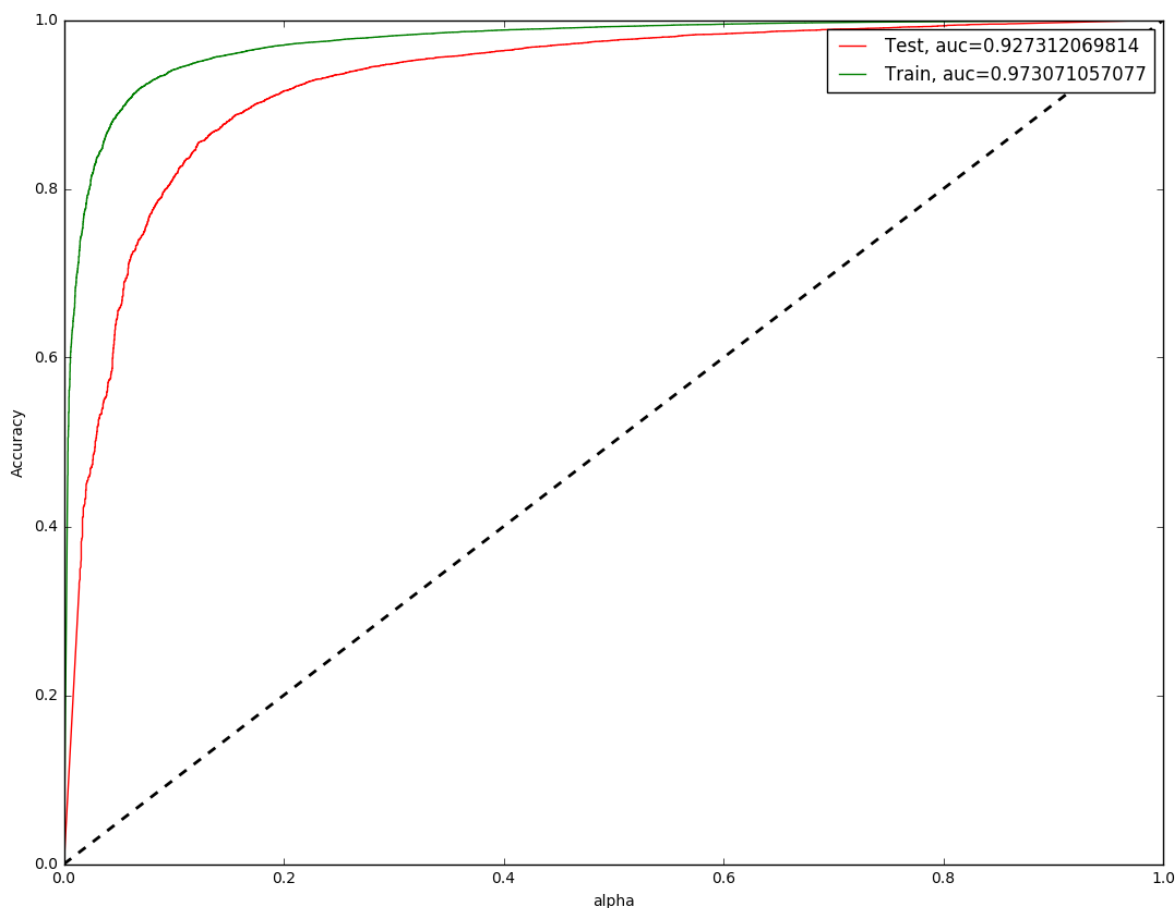
0.973071057077

In [205]:

```
import pylab
plt.figure(figsize=(13, 10))
plt.plot([0,1], [0,1], color='black', lw=2, linestyle='--')
plt.plot(fpr_test,tpr_test,label="Test, auc="+str(bow_test_auc), color = 'red')
plt.plot(fpr_train,tpr_train,label="Train, auc="+str(bow_train_auc), color = 'green')

plt.xlabel('alpha')
plt.ylabel('Accuracy')
plt.legend()

plt.show()
```



In [206]:

```

from sklearn.metrics import classification_report, confusion_matrix
bow_train_conf_matrix = confusion_matrix(Y_train, bow_train_conf)
bow_test_conf_matrix = confusion_matrix(Y_test, bow_test_conf)
class_report = classification_report(Y_test, bow_test_conf)
print(bow_test_conf_matrix)
print(class_report)

```

```

[[ 1878   798]
 [  900 16424]]

```

	precision	recall	f1-score	support
0	0.68	0.70	0.69	2676
1	0.95	0.95	0.95	17324
avg / total	0.92	0.92	0.92	20000

In [207]:

```

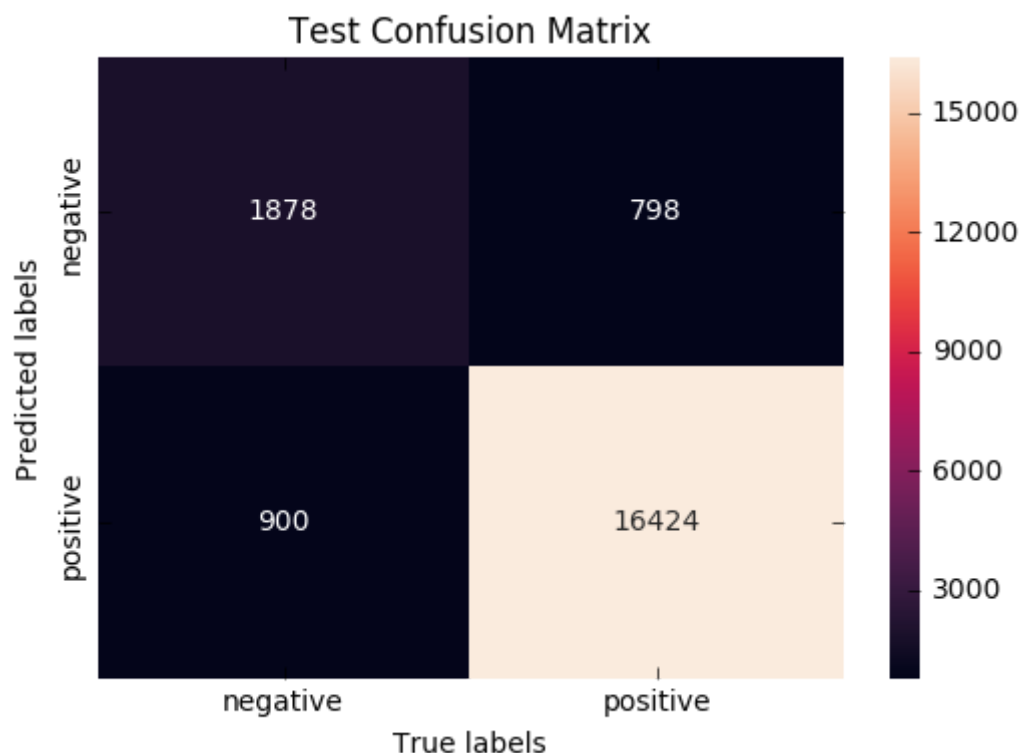
ax= plt.subplot()
sns.heatmap(bow_test_conf_matrix, annot=True, ax = ax, fmt='g')

ax.set_ylabel('Predicted labels')
ax.set_xlabel('True labels')
ax.set_title('Test Confusion Matrix')
ax.xaxis.set_ticklabels(['negative', 'positive'])
ax.yaxis.set_ticklabels(['negative', 'positive'])

```

Out[207]:

```
[<matplotlib.text.Text at 0xc62814a8>, <matplotlib.text.Text at 0xbb2f4e10>]
```



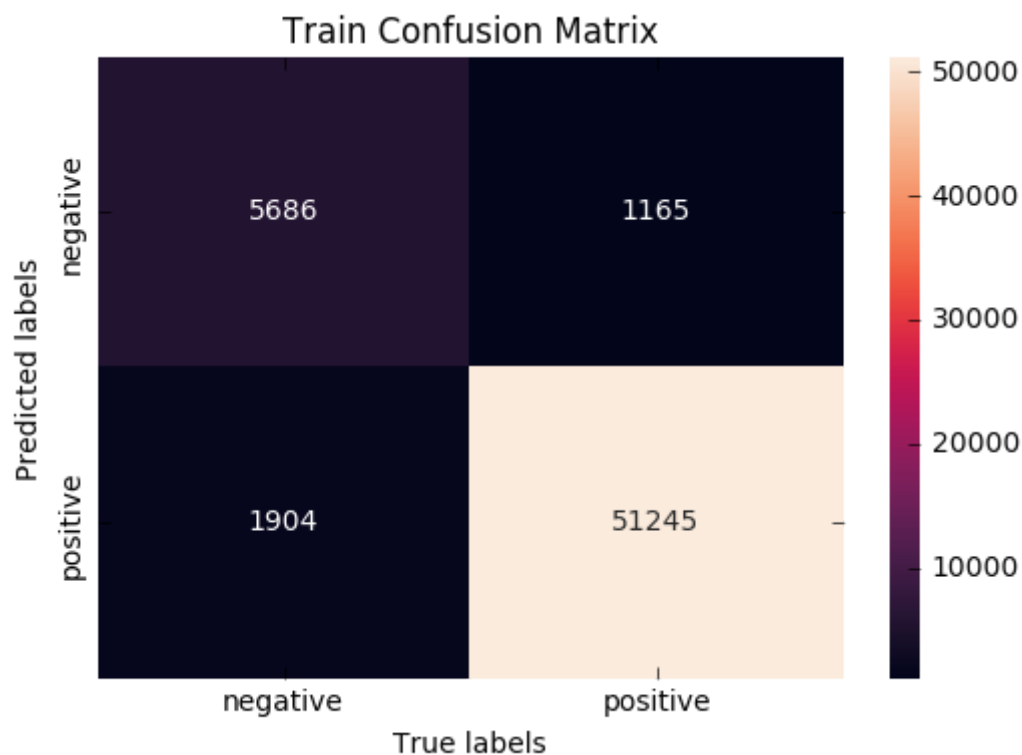
In [208]:

```
ax= plt.subplot()
sns.heatmap(bow_train_conf_matrix, annot=True, ax = ax, fmt='g')

ax.set_ylabel('Predicted labels')
ax.set_xlabel('True labels')
ax.set_title('Train Confusion Matrix')
ax.xaxis.set_ticklabels(['negative', 'positive'])
ax.yaxis.set_ticklabels(['negative', 'positive'])
```

Out[208]:

```
[<matplotlib.text.Text at 0xbbc1a3c8>, <matplotlib.text.Text at 0x1084a65f8>]
```



Naive Bayes on TF-IDF

In [213]:

```
import pickle
# with open(r"/content/gdrive/My Drive/Colab Notebooks/Assignment 4/tf_idf.pkl", "rb") as i
with open(r"tf_idf.pkl", "rb") as input_file:
    tfidf_dict = pickle.load(input_file)
```

In [216]:

```
from sklearn.naive_bayes import MultinomialNB
from tqdm import tqdm

tfidf_auc_train = []
tfidf_auc_cv = []
auc_test = []

for alpha_value in [ 0.00001, 0.0001, 0.001, 0.01, 0.1, 1, 10, 100, 1000, 10000, 100000]:
    nb_tfidf = MultinomialNB(alpha=alpha_value)
    nb_tfidf.fit(tfidf_dict['train_tf_idf'], Y_train)

    train_proba = nb_tfidf.score(tfidf_dict['train_tf_idf'], Y_train)
    tfidf_auc_train.append(train_proba)
    cv_proba = nb_tfidf.score(tfidf_dict['cv_tf_idf'], Y_val)
    tfidf_auc_cv.append(cv_proba)

f = most_informative_feature_for_binary_classification(tf_idf_vect, nb_tfidf)
```

Top 10 features for negative class

	Coeff	Labels	Negative features
0	-10.488483	0	cancel order
1	-10.488483	0	give zero
2	-10.488483	0	lesson learned
3	-10.488483	0	no stars
4	-10.488483	0	threw rest
5	-10.488483	0	worst coffee
6	-10.488483	0	worst tasting
7	-10.488482	0	false advertising
8	-10.488482	0	hopes product
9	-10.488481	0	zero stars

Top 10 features for positive class

	Coeff	Labels	Negative features
0	-10.474219	1	not
1	-10.475047	1	great
2	-10.476671	1	tea
3	-10.477038	1	good
4	-10.479447	1	like
5	-10.479973	1	love
6	-10.480086	1	best
7	-10.480207	1	product
8	-10.480700	1	taste
9	-10.480751	1	coffee

In [217]:

```
alpha_list = [ 0.00001, 0.0001, 0.001, 0.01, 0.1, 1, 10, 100, 1000, 10000, 100000]
train_alpha_dict = dict(zip(alpha_list, tfidf_auc_train))
val_alpha_dict = dict(zip(alpha_list, tfidf_auc_cv))
print(train_alpha_dict)
print(val_alpha_dict)
```

```
{0.1: 0.95265, 1: 0.9061666666666668, 100: 0.8858166666666667, 1000: 0.8858166666666667, 10000: 0.8858166666666667, 1e-05: 0.9609833333333333, 0.0001: 0.9607499999999999, 10: 0.8856833333333338, 100000: 0.8858166666666667, 0.01: 0.9593833333333337, 0.001: 0.96035000000000004}
{0.1: 0.9186499999999997, 1: 0.8839500000000001, 100: 0.8628000000000001, 1000: 0.8628000000000001, 10000: 0.8628000000000001, 1e-05: 0.9030000000000002, 0.0001: 0.9054999999999997, 10: 0.8627000000000002, 100000: 0.8628000000000001, 0.01: 0.9147999999999995, 0.001: 0.9099500000000004}
```

In [218]:

```
tfidf_best_alpha = max(val_alpha_dict, key=val_alpha_dict.get)
tfidf_best_alpha
```

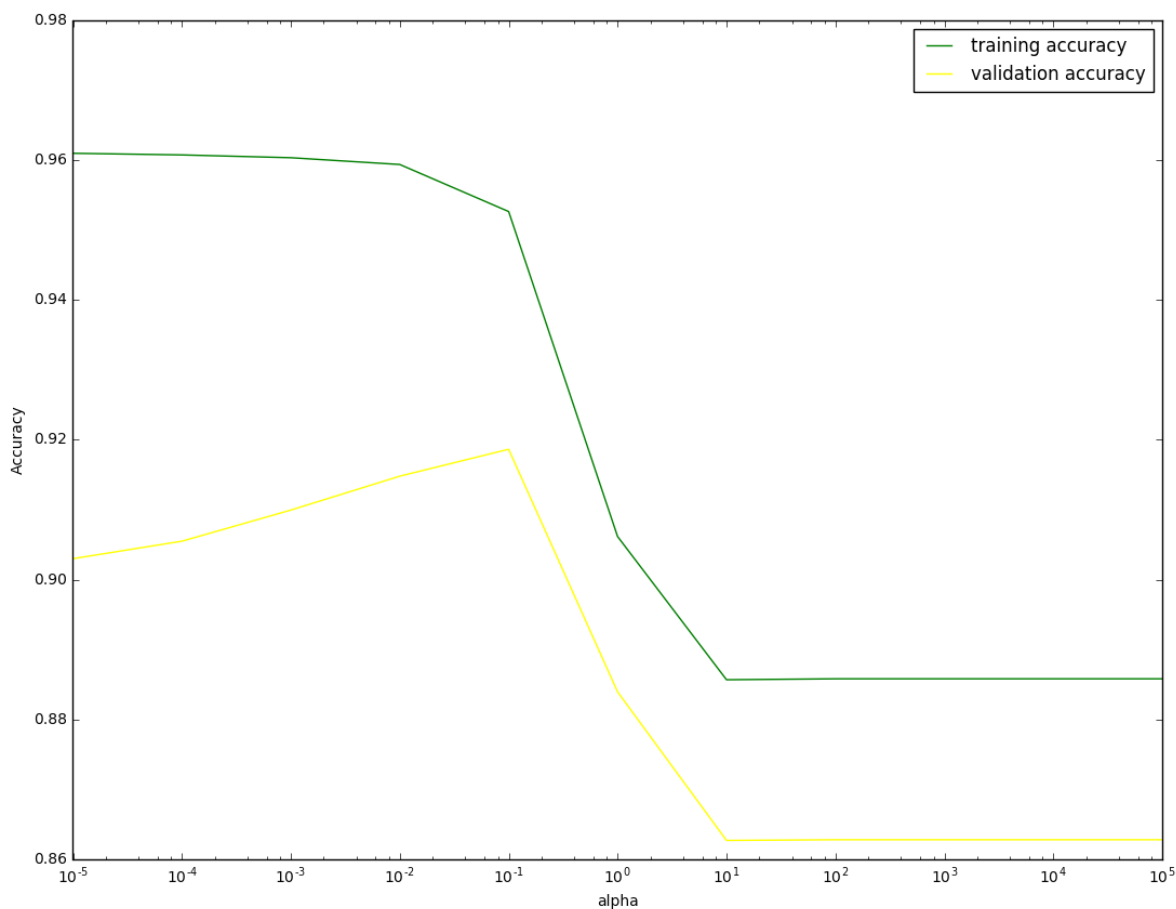
Out[218]:

0.1

In [219]:

```
import pylab
plt.figure(figsize=(13, 10))
# max_idx = auc_train.index(max(tfidf_auc_train))
neighbors_settings = [ 0.00001, 0.0001, 0.001, 0.01, 0.1, 1, 10, 100, 1000, 10000, 100000]
plt.plot(neighbors_settings, tfidf_auc_train, label="training accuracy", color='green')
plt.plot(neighbors_settings, tfidf_auc_cv, label="validation accuracy", color='yellow')
# plt.plot(neighbors_settings, auc_test, label="test accuracy", color='red')
plt.xlabel('alpha')
plt.ylabel('Accuracy')
plt.legend()
plt.xscale('log')

plt.show()
```



In [220]:

```
nb = MultinomialNB(alpha=tfidf_best_alpha)
nb.fit(tfidf_dict['train_tf_idf'], Y_train)
tfidf_test_predict = nb.predict_proba(tfidf_dict['test_tf_idf'])
tfidf_train_predict = nb.predict_proba(tfidf_dict['train_tf_idf'])
tfidf_train_conf = nb.predict(tfidf_dict['train_tf_idf'])
tfidf_test_conf = nb.predict(tfidf_dict['test_tf_idf'])
tfidf_test_predict
```

Out[220]:

```
array([[ 0.00777046,  0.99222954],
       [ 0.61995316,  0.38004684],
       [ 0.00291888,  0.99708112],
       ...,
       [ 0.13856742,  0.86143258],
       [ 0.00127726,  0.99872274],
       [ 0.52000291,  0.47999709]])
```

In [221]:

```
print(len(tfidf_auc_train))
print(len(tfidf_train_predict))
```

```
11
60000
```

In [222]:

```
tfidf_fpr_train, tfidf_tpr_train, tfidf_thresholds_train = roc_curve(Y_train, tfidf_train_predict)
tfidf_fpr_test, tfidf_tpr_test, tfidf_thresholds_test = roc_curve(Y_test, tfidf_test_predict)
tfidf_test_auc = auc(tfidf_fpr_test, tfidf_tpr_test)
tfidf_train_auc = auc(tfidf_fpr_train, tfidf_tpr_train)
print(tfidf_test_auc)
print(tfidf_train_auc)
```

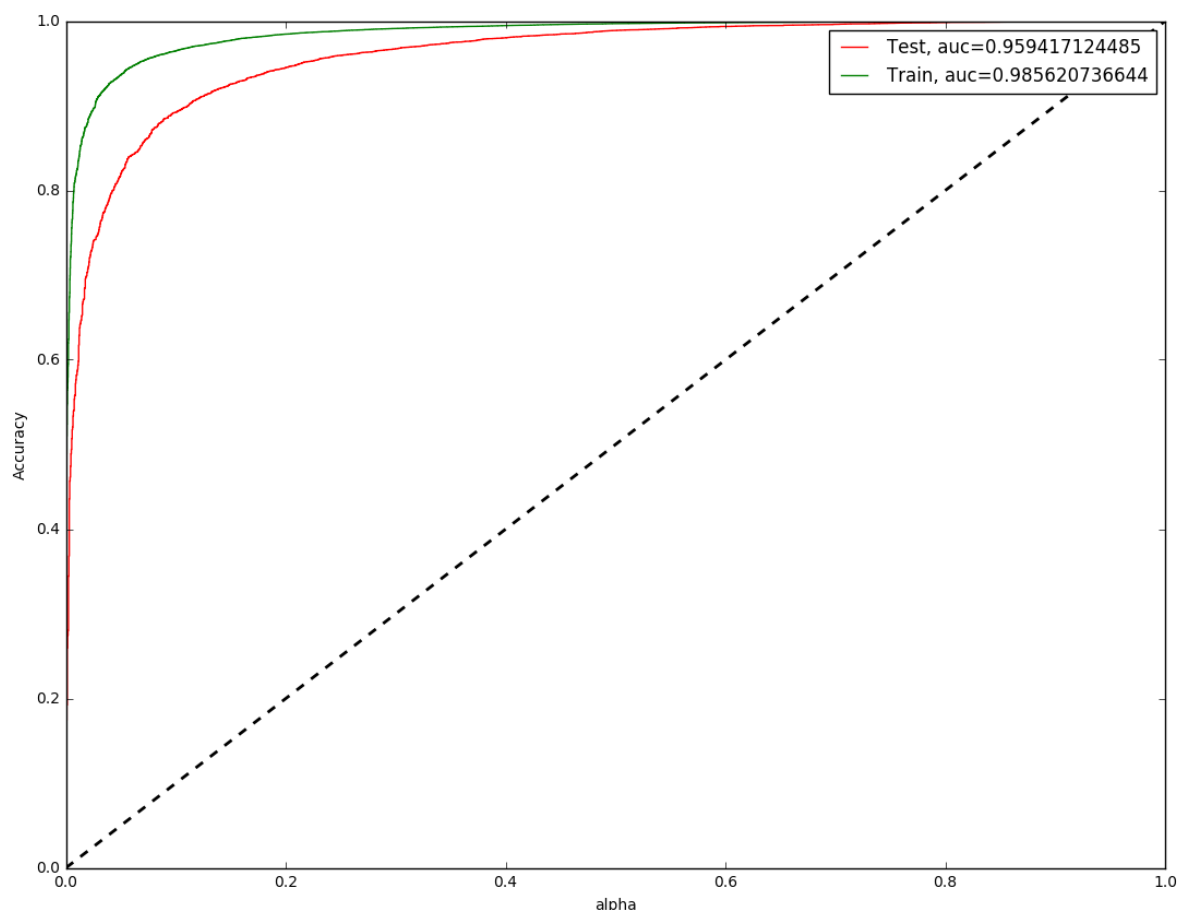
```
0.959417124485
0.985620736644
```

In [223]:

```
import pylab
plt.figure(figsize=(13, 10))
plt.plot([0,1], [0,1], color='black', lw=2, linestyle='--')
plt.plot(tfidf_fpr_test,tfidf_tpr_test,label="Test, auc="+str(tfidf_test_auc), color = 'red')
plt.plot(tfidf_fpr_train,tfidf_tpr_train,label="Train, auc="+str(tfidf_train_auc), color = 'green')

plt.xlabel('alpha')
plt.ylabel('Accuracy')
plt.legend()

plt.show()
```



In [224]:

```
from sklearn.metrics import classification_report, confusion_matrix
tfidf_train_conf_matrix = confusion_matrix(Y_train, tfidf_train_conf)
tfidf_test_conf_matrix = confusion_matrix(Y_test, tfidf_test_conf)
class_report = classification_report(Y_test, tfidf_test_conf)
print(tfidf_test_conf_matrix)
print(class_report)
```

```
[[ 1251  1425]
 [   168 17156]]
```

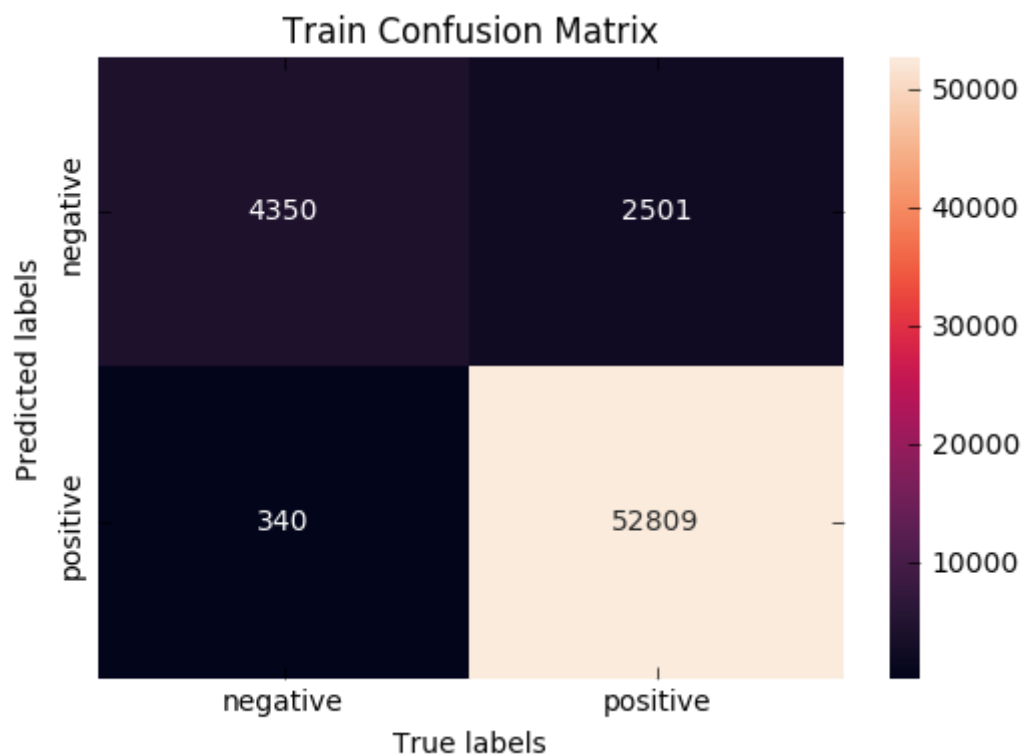
	precision	recall	f1-score	support
0	0.88	0.47	0.61	2676
1	0.92	0.99	0.96	17324
avg / total	0.92	0.92	0.91	20000

In [225]:

```
ax= plt.subplot()  
sns.heatmap(tfidf_train_conf_matrix, annot=True, ax = ax, fmt='g')  
ax.set_ylabel('Predicted labels')  
ax.set_xlabel('True labels')  
ax.set_title('Train Confusion Matrix')  
ax.xaxis.set_ticklabels(['negative', 'positive'])  
ax.yaxis.set_ticklabels(['negative', 'positive'])
```

Out[225]:

[<matplotlib.text.Text at 0xc4134cf8>, <matplotlib.text.Text at 0xfeed97b70>]

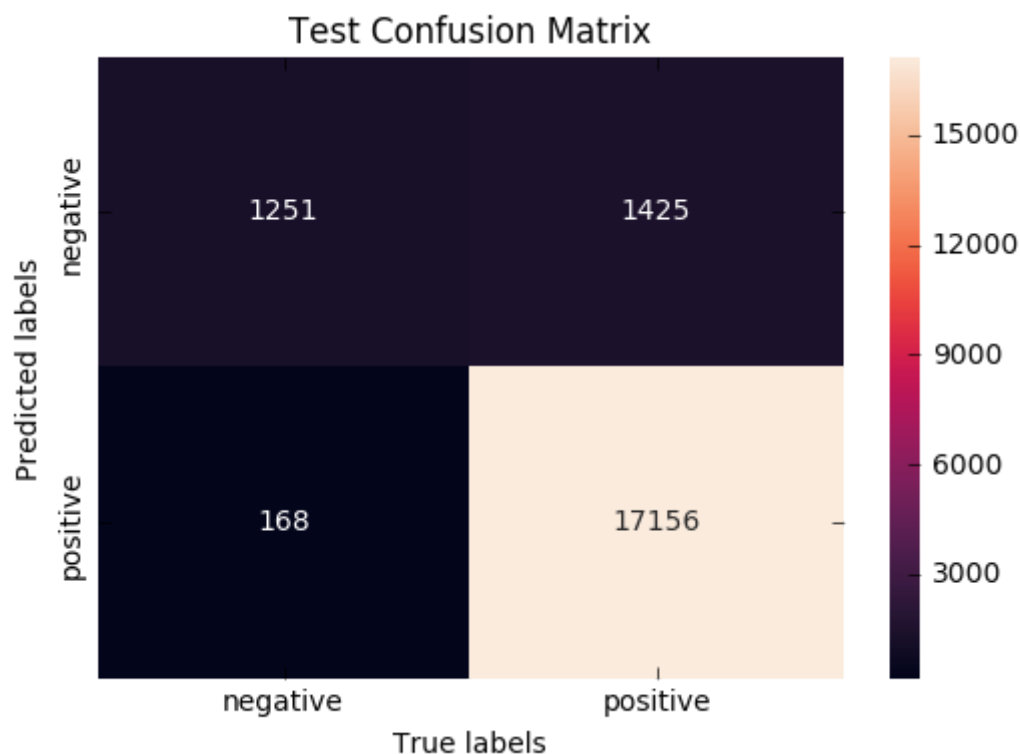


In [226]:

```
ax= plt.subplot()
sns.heatmap(tfidf_test_conf_matrix, annot=True, ax = ax, fmt='g')
ax.set_ylabel('Predicted labels')
ax.set_xlabel('True labels')
ax.set_title('Test Confusion Matrix')
ax.xaxis.set_ticklabels(['negative', 'positive'])
ax.yaxis.set_ticklabels(['negative', 'positive'])
```

Out[226]:

```
[<matplotlib.text.Text at 0xb252c2e8>, <matplotlib.text.Text at 0xa72987b8>]
```



In [227]:

```
from prettytable import PrettyTable

x = PrettyTable()
x.field_names = ["Vectorizer", "alpha", "Train", "Cross validation", "Test"]

x.add_row(["BoW", bow_best_alpha, bow_train_auc, max(bow_auc_cv), bow_test_auc])
x.add_row(["Tf-idf", tfidf_best_alpha, tfidf_train_auc, max(tfidf_auc_cv), tfidf_test_auc])
print(x)
```

```
+-----+-----+-----+-----+-----+
| Vectorizer | alpha | Train | Cross validation | Test |
+-----+-----+-----+-----+-----+
| BoW       | 0.1   | 0.973071057077 | 0.9166 | 0.927312069814 |
| Tf-idf    | 0.1   | 0.985620736644 | 0.91865 | 0.959417124485 |
+-----+-----+-----+-----+-----+
```

Steps taken to increase accuracy:

- i. Summary and Text columns are appended in single column
- ii. length of words is taken from appended column and stacked with sparse matrix

Observations:

- i. Accuracy increased around 2% for each vectorizer.