3.6 Featurizing text data with tfidf weighted word-vectors

In [40]:
```
import pandas as pd
import matplotlib.pyplot as plt
import re
import time
import warnings
import numpy as np
from nltk.corpus import stopwords
from sklearn.preprocessing import normalize
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.feature_extraction.text import TfidfVectorizer
warnings.filterwarnings("ignore")
import sys
import os
import pandas as pd
import numpy as np
from tqdm import tqdm
import pickle

# exctract word2vec vectors
# https://github.com/explosion/spaCy/issues/1721
# http://landinghub.visualstudio.com/visual-cpp-build-tools
import spacy
from nltk.stem import PorterStemmer
from bs4 import BeautifulSoup
from nltk import word_tokenize
from scipy import sparse
from sklearn.model_selection import train_test_split
from collections import Counter
```

In [41]:
```
# avoid decoding problems
df = pd.read_csv("train.csv")

# encode questions to unicode
# https://stackoverflow.com/a/6812069
# ----------------- python 2 ---------------------
# df['question1'] = df['question1'].apply(lambda x: unicode(str(x),"utf-8"))
# df['question2'] = df['question2'].apply(lambda x: unicode(str(x),"utf-8"))
# ----------------- python 3 ---------------------
df['question1'] = df['question1'].apply(lambda x: str(x))
df['question2'] = df['question2'].apply(lambda x: str(x))
```

In [42]:
```
df.head()
```

Out[42]:

	id	qid1	qid2	question1	question2	is_duplicate
0	0	1	2	What is the step by step guide to invest in sh...	What is the step by step guide to invest in sh...	0
1	1	3	4	What is the story of Kohinoor (Koh-i-Noor) Dia...	What would happen if the Indian government sto...	0
2	2	5	6	How can I increase the speed of my internet co...	How can Internet speed be increased by hacking...	0
3	3	7	8	Why am I mentally very lonely? How can I solve...	Find the remainder when [math]23^{24}[/math] i...	0
4	4	9	10	Which one dissolve in water quikly sugar, salt...	Which fish would survive in salt water?	0

In [43]:
```
# To get the results in 4 decemal points
SAFE_DIV = 0.0001

STOP_WORDS = stopwords.words("english")


def preprocess(x):
    x = str(x).lower()
    x = x.replace(",000,000", "m").replace(",000", "k").replace("′", "'").replace("’", "'")\
                           .replace("won't", "will not").replace("cannot", "can not").replace("can't", "can not")\
                           .replace("n't", " not").replace("what's", "what is").replace("it's", "it is")\
                           .replace("'ve", " have").replace("i'm", "i am").replace("'re", " are")\
                           .replace("he's", "he is").replace("she's", "she is").replace("'s", " own")\
                           .replace("%", " percent ").replace("₹", " rupee ").replace("$", " dollar ")\
                           .replace("€", " euro ").replace("'ll", " will")
    x = re.sub(r"([0-9]+)000000", r"\1m", x)
    x = re.sub(r"([0-9]+)000", r"\1k", x)


    porter = PorterStemmer()
    pattern = re.compile('\W')

    if type(x) == type(''):
        x = re.sub(pattern, ' ', x)


    if type(x) == type(''):

        example1 = BeautifulSoup(x)
        x = example1.get_text()
        word=word_tokenize(x)
        lit=[]
        for i in word:
            lit.append(porter.stem(i))
            u=' '.join(lit)
            u=str(u)

    return x
```

In [5]:
```
#prepro_features_train.csv (Simple Preprocessing Feartures)
#nlp_features_train.csv (NLP Features)
if os.path.isfile('nlp_features_train.csv'):
    dfnlp = pd.read_csv("nlp_features_train.csv",encoding='latin-1')
else:
    print("download nlp_features_train.csv from drive or run previous notebook")

if os.path.isfile('df_fe_without_preprocessing_train.csv'):
    dfppro = pd.read_csv("df_fe_without_preprocessing_train.csv",encoding='latin-1')
else:
    print("download df_fe_without_preprocessing_train.csv from drive or run previous notebook")
```

In [6]:
```
df1 = dfnlp.drop(['qid1','qid2','question1','question2','is_duplicate'],axis=1)
df2 = dfppro.drop(['qid1','qid2','question1','question2','is_duplicate'],axis=1)
```

In [7]:
```
df_f=df1.merge(df2 , on='id', how= 'left')
```

In [8]:
```
df_f.shape
```

Out[8]: (404290, 27)

In [10]:
```
df = df.merge(df_f , on='id', how='left')
```

In [11]:
```
df.shape
```

Out[11]: (404290, 32)

In [15]:
```
df= df[0:100000]
y_true = df['is_duplicate']
```

In [16]:
```
df.drop('is_duplicate', axis=1, inplace=True)
```

In [17]:
```
df.shape
```

Out[17]: (100000, 31)

In [18]:
```
df["question1"] = df["question1"].fillna("").apply(preprocess)
df["question2"] = df["question2"].fillna("").apply(preprocess)
```

In [19]:
```
que= np.dstack([df['question1'],df['question2']])

ques=[]
for i in que[0]:
    i=' '.join(i)
    ques.append(i)

questions = pd.DataFrame(ques , columns=['questions'])
```

In [20]:
```
questions['id']= df['id']
```

In [21]:
```
df= df.merge(questions ,on='id', how ='left')
```

In [22]:
```
df.head()
```

Out[22]:

	id	qid1	qid2	question1	question2	cwc_min	cwc_max	csc_min	csc_max	ctc_min	...	q1len	q2len	q1_n_words	q2_n_words	word_Com
0	0	1	2	what is the step by step guide to invest in sh...	what is the step by step guide to invest in sh...	0.999980	0.833319	0.999983	0.999983	0.916659	...	66	57	14	12	10.0
1	1	3	4	what is the story of kohinoor koh i noor dia...	what would happen if the indian government sto...	0.799984	0.399996	0.749981	0.599988	0.699993	...	51	88	8	13	4.0
2	2	5	6	how can i increase the speed of my internet co...	how can internet speed be increased by hacking...	0.399992	0.333328	0.399992	0.249997	0.399996	...	73	59	14	10	4.0
3	3	7	8	why am i mentally very lonely how can i solve...	find the remainder when math 23 24 math i...	0.000000	0.000000	0.000000	0.000000	0.000000	...	50	65	11	9	0.0
4	4	9	10	which one dissolve in water quikly sugar salt...	which fish would survive in salt water	0.399992	0.199998	0.999950	0.666644	0.571420	...	76	39	13	7	2.0

5 rows × 32 columns

In [29]:
```
X_train , X_test , y_train , y_test = train_test_split(df , y_true ,stratify =y_true , test_size=0.3)
```

In [30]:
```
X_train ,X_cv ,y_train ,y_cv = train_test_split(X_train , y_train , stratify=y_train ,test_size=0.3)
```

In [31]:
```
print("Number of data points in train data :",X_train.shape)
print("Number of data points in train data :",X_cv.shape)
print("Number of data points in test data :",X_test.shape)
```

```
Number of data points in train data : (49000, 32)
Number of data points in train data : (21000, 32)
Number of data points in test data : (30000, 32)
```

In [32]:
```
print("-"*10, "Distribution of output variable in train data", "-"*10)
train_distr = Counter(y_train)
train_len = len(y_train)
print("Class 0: ",int(train_distr[0])/train_len,"Class 1: ", int(train_distr[1])/train_len)

print("-"*10, "Distribution of output variable in test data", "-"*10)
test_distr = Counter(y_test)
test_len = len(y_test)
print("Class 0: ",int(test_distr[0])/test_len, "Class 1: ",int(test_distr[1])/test_len)

print("-"*10, "Distribution of output variable in cv data", "-"*10)
cv_distr = Counter(y_cv)
cv_len = len(y_cv)
print("Class 0: ",int(cv_distr[0])/cv_len, "Class 1: ",int(cv_distr[1])/cv_len)
```

```
---------- Distribution of output variable in train data ----------
Class 0:  0.6274489795918368 Class 1:  0.3725510204081633
---------- Distribution of output variable in test data ----------
Class 0:  0.6274666666666666 Class 1:  0.3725333333333333
---------- Distribution of output variable in cv data ----------
Class 0:  0.6274761904761905 Class 1:  0.3725238095238953
```

In [33]:
```
from sklearn.feature_extraction.text import TfidfVectorizer

tfidf = TfidfVectorizer(lowercase=False, max_features= 5000 )

train_tfidf= tfidf.fit_transform(X_train['questions'].values)
cv_tfidf= tfidf.transform(X_cv['questions'].values)
test_tfidf =tfidf.transform(X_test['questions'].values)

tfidf_feats=tfidf.get_feature_names()

print("The shape of train tfidf ",train_tfidf.shape)
print("The shape of cv tfidf ",cv_tfidf.shape)
print("The shape of test tfidf ",test_tfidf.shape)
```

```
The shape of train tfidf  (49000, 5000)
The shape of cv tfidf  (21000, 5000)
The shape of test tfidf  (30000, 5000)
```

In [34]:
```
X_train = X_train.drop(['id','qid1','qid2','question1','question2','questions'],axis=1)
X_cv = X_cv.drop(['id','qid1','qid2','question1','question2','questions'],axis=1)
X_test = X_test.drop(['id','qid1','qid2','question1','question2','questions'],axis=1)
```

In [35]:
```
for colu in X_train:
    column=np.array(X_train[colu][0:train_tfidf.shape[0]])
    train_tfidf=sparse.hstack((train_tfidf,column[:,None]))

for colu in X_cv:
    column=np.array(X_cv[colu][0:cv_tfidf.shape[0]])
    cv_tfidf=sparse.hstack((cv_tfidf,column[:,None]))

for colu in X_test:
    column=np.array(X_test[colu][0:test_tfidf.shape[0]])
    test_tfidf=sparse.hstack((test_tfidf,column[:,None]))
```

In [36]:
```
print("The shape of final train data is ",train_tfidf.shape)
print("The shape of final cv data is ",cv_tfidf.shape)
print("The shape of final test data is ",test_tfidf.shape)
```

```
The shape of final train data is  (49000, 5026)
The shape of final cv data is  (21000, 5026)
The shape of final test data is  (30000, 5026)
```

In [37]:
```
sparse.save_npz('train_tfidf',train_tfidf)              #storing the sparse matrix
sparse.save_npz('cv_tfidf',cv_tfidf)
sparse.save_npz('test_tfidf',test_tfidf)
```

In [39]:
```
y_train.to_pickle('y_train')                            #storing the true labels
y_cv.to_pickle('y_cv')
y_test.to_pickle('y_test')
```