3.6 Featurizing text data with tfidf weighted

In [1]:

```
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
import tqdm

# 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
```

In [2]:

```
# 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 [3]:

```
df.head()
```

Out[3]:

	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 [4]:

```
# Take 100k datasample
df = df.sample(n=100000, random_state=1)
df.shape
```

Out[4]:

```
(100000, 6)
```

In [5]:

```
# To get the results in 4 decemal points
SAFE_DIV = 0.0001

STOP_WORDS = stopwords.words("english")


def preprocess(x):
    # Convert into lowercase
    x = str(x).lower()
    # Replacing some the string into other string
    # For eg -> 1,000 replaced with 1k, he's replaced with he is
    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")
    # If an regular expression where appear x'000000' then it replaced with xm where x is any number
    x = re.sub(r"([0-9]+)000000", r"\1m", x)
    # If an regular expression where appear x'000' then it replaced with xk where x is any number
    x = re.sub(r"([0-9]+)000", r"\1k", x)

    # Stemming algorithm with NLTK
    # Ref : https://www.geeksforgeeks.org/python-stemming-words-with-nltk/
    porter = PorterStemmer()
    # This matches any non-alphanumeric character [^a-zA-Z0-9]
    # Ref : https://scotch.io/tutorials/an-introduction-to-regex-in-python
    pattern = re.compile('\W')

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

    if type(x) == type(''):
        x = porter.stem(x)
        # Ref : https://www.crummy.com/software/BeautifulSoup/bs4/doc/
        example1 = BeautifulSoup(x)
        # New only text (rest of the part will automatically neglect)
        x = example1.get_text()

    y = [i for i in x.split() if i not in STOP_WORDS]
    y = ' '.join(y)
    return y
```

In [6]:

```
clean_q1 = []
clean_q2 = []
for i in tqdm.tqdm_notebook(df['question1'].values):
    clean_q1.append(preprocess(i))
for i in tqdm.tqdm_notebook(df['question2'].values):
    clean_q2.append(preprocess(i))
```

In [7]:

```
df['question1'] = clean_q1
df['question2'] = clean_q2
df.head()
```

Out[7]:

	id	qid1	qid2	question1	question2	is_duplicate
237030	237030	33086	348102	stop playing video games	stop playing video games child	0
247341	247341	73272	8624	better donald trump hillary clinton	hillary clinton better choice donald trump	1
246425	246425	359482	359483	think chance sometime 21st century another maj...	think another world war nuclear war 21st century	1
306985	306985	1357	47020	many questions posted quora easily answered us...	people write questions quora could answered qu...	1
225863	225863	334315	334316	even movie ever rated 10 10 imdb	10 10 movies	0

In [8]:

```
# Split the data into 70,30 train and test data
from sklearn.model_selection import train_test_split
tr, ts = train_test_split(df, test_size=0.3, random_state=1, stratify=df['is_duplicate'].values)
```

In [9]:

```
tr.shape, ts.shape
```

Out[9]:

```
((70000, 6), (30000, 6))
```

In [10]:

```
tr.head()
```

Out[10]:

	id	qid1	qid2	question1	question2	is_duplicate
149483	149483	235456	235457	translate android	would translate 螳螂捕蝉 黄雀在后	0
146085	146085	179014	230839	pet bird trained live without cage fly away	airports keep birds away	0
337094	337094	44878	204218	best way teach child swim	teach kid swim	1
115033	115033	187657	187658	add location business page facebook	facebook page check place best location servic...	0
190104	190104	289081	289082	purpose roman colosseum	purpose colosseum serve	1

In [11]:

```
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.feature_extraction.text import CountVectorizer
# merge texts
questions = list(tr['question1']) + list(tr['question2'])

# Fit the Tfidf vectorizer on train data
tfidf = TfidfVectorizer(lowercase=False, min_df=10, max_features=4000)
tfidf.fit(questions)
```

Out[11]:

```
TfidfVectorizer(analyzer='word', binary=False, decode_error='strict',
        dtype=<class 'numpy.float64'>, encoding='utf-8', input='content',
        lowercase=False, max_df=1.0, max_features=4000, min_df=10,
        ngram_range=(1, 1), norm='l2', preprocessor=None, smooth_idf=True,
        stop_words=None, strip_accents=None, sublinear_tf=False,
        token_pattern='(?u)\\b\\w\\w+\\b', tokenizer=None, use_idf=True,
        vocabulary=None)
```

In [12]:

```
# transform tfidf on question1 and question2 on train data
```

```
tr_q1_feats_m = tfidf.transform(tr['question1'].values)
tr_q2_feats_m = tfidf.transform(tr['question2'].values)

# transform tfidf on question1 and question2 on test data
ts_q1_feats_m = tfidf.transform(ts['question1'].values)
ts_q2_feats_m = tfidf.transform(ts['question2'].values)
```

In [13]:

```
tr_q1_feats_m.shape, tr_q2_feats_m.shape
```

Out[13]:

```
((70000, 4000), (70000, 4000))
```

In [14]:

```
#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_basicfe_train.csv'):
    dfppro = pd.read_csv("df_basicfe_train.csv",encoding='latin-1')
else:
    print("download df_basicfe_train.csv from drive or run previous notebook")
```

In [15]:

```
# Take 100k and split in same proportion
# Random state parameter give the good idea that if you give random state to any value, and recompile a
gain and again,
# the result will always the same.

df1 = dfnlp.sample(n=100000, random_state=1)
df2 = dfppro.sample(n=100000, random_state=1)
df1.shape, df2.shape
```

Out[15]:

```
((100000, 21), (100000, 17))
```

In [16]:

```
df_tr1, df_ts1 = train_test_split(df1, test_size=0.3, random_state=1, stratify=df1['is_duplicate'].valu
es)
df_tr2, df_ts2 = train_test_split(df2, test_size=0.3, random_state=1, stratify=df2['is_duplicate'].valu
es)
df_tr1.shape, df_tr2.shape, df_ts1.shape, df_ts2.shape
```

Out[16]:

```
((70000, 21), (70000, 17), (30000, 21), (30000, 17))
```

In [17]:

```
df_tr1 = df_tr1.drop(['qid1','qid2','question1','question2'],axis=1)
df_ts1 = df_ts1.drop(['qid1','qid2','question1','question2'],axis=1)
df_tr2 = df_tr2.drop(['qid1','qid2','question1','question2','is_duplicate'],axis=1)
df_ts2 = df_ts2.drop(['qid1','qid2','question1','question2','is_duplicate'],axis=1)
```

In [18]:

```
# Just take tfidf w2v feature only and remove others
df3 = tr.drop(['qid1','qid2','question1','question2','is_duplicate'],axis=1)
```

```
# Store tfidf w2v of question1 train data
df3_q1 = pd.DataFrame(tr_q1_feats_m.toarray(), index= df3.index)

# Store tfidf w2v of question2 train data
df3_q2 = pd.DataFrame(tr_q2_feats_m.toarray(), index= df3.index)

# Just take tfidf w2v feature only and remove others
df3 = ts.drop(['qid1','qid2','question1','question2','is_duplicate'],axis=1)

# Store tfidf w2v of question1 train data
df4_q1 = pd.DataFrame(ts_q1_feats_m.toarray(), index= df3.index)

# Store tfidf w2v of question2 train data
df4_q2 = pd.DataFrame(ts_q2_feats_m.toarray(), index= df3.index)
```

In [19]:

```
# dataframe of advance nlp feature of train data
df_tr1.head()
```

Out[19]:

	id	is_duplicate	cwc_min	cwc_max	csc_min	csc_max	ctc_min	ctc_max	last_word_eq	first_word_eq	abs_len_dif
149483	149483	0	0.499975	0.249994	0.499975	0.249994	0.333328	0.333328	0.0	1.0	0.0
146085	146085	0	0.249994	0.124998	0.000000	0.000000	0.166664	0.066666	1.0	0.0	9.0
337094	337094	1	0.666644	0.399992	0.399992	0.333328	0.499994	0.333331	1.0	0.0	4.0
115033	115033	0	0.599988	0.374995	0.285710	0.222220	0.416663	0.238094	0.0	0.0	9.0
190104	190104	1	0.666644	0.666644	0.666644	0.499988	0.666656	0.571420	0.0	1.0	1.0

In [20]:

```
# dataframe of advance nlp feature of test data
df_ts1.head()
```

Out[20]:

	id	is_duplicate	cwc_min	cwc_max	csc_min	csc_max	ctc_min	ctc_max	last_word_eq	first_word_eq	abs_len_dif
303972	303972	0	0.833319	0.555549	0.999986	0.636358	0.857137	0.599997	0.0	0.0	6.0
72206	72206	1	0.499992	0.499992	0.333322	0.199996	0.363633	0.333331	1.0	0.0	1.0
106335	106335	1	0.999983	0.857131	0.999986	0.874989	0.999992	0.866661	0.0	1.0	2.0
268194	268194	0	0.583328	0.538457	0.636358	0.583328	0.499998	0.466665	0.0	0.0	2.0
33364	33364	0	0.333322	0.166664	0.000000	0.000000	0.166664	0.090908	0.0	0.0	5.0

In [21]:

```
# Dataframe of basic feature of train data
df_tr2.head()
```

Out[21]:

	id	freq_qid1	freq_qid2	q1len	q2len	q1_n_words	q2_n_words	word_Common	word_Total	word_share	freq_q1+q2	f
149483	149483	1	1	30	37	6	6	2.0	12.0	0.166667	2	
146085	146085	2	1	66	32	15	6	1.0	20.0	0.050000	3	
337094	337094	3	4	50	34	12	8	4.0	19.0	0.210526	7	
115033	115033	1	1	56	96	12	19	4.0	27.0	0.148148	2	
190104	190104	1	1	42	38	7	6	4.0	13.0	0.307692	2	

In [22]:

```
# Dataframe of basic feature of train data
df_ts2.head()
```

Out[22]:

	id	freq_qid1	freq_qid2	q1len	q2len	q1_n_words	q2_n_words	word_Common	word_Total	word_share	freq_q1+q2
303972	303972	1	1	60	102	13	19	11.0	31.0	0.354839	2
72206	72206	3	2	61	59	10	11	3.0	19.0	0.157895	5
106335	106335	1	1	79	68	15	12	10.0	27.0	0.370370	2
268194	268194	1	1	133	140	26	30	11.0	49.0	0.224490	2
33364	33364	1	1	32	58	6	11	1.0	17.0	0.058824	2

In [23]:

```
# Questions 1 tfidf weighted word2vec
df3_q1.head()
```

Out[23]:

	0	1	2	3	4	5	6	7	8	9	...	3990	3991	3992	3993	3994	3995	3996	3997	3998	3999
149483	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
146085	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
337094	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
115033	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
190104	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

5 rows × 4000 columns

In [24]:

```
df4_q1.head()
```

Out[24]:

	0	1	2	3	4	5	6	7	8	9	...	3990	3991	3992	3993	3994	3995	3996	3997	3998	3999
303972	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
72206	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
106335	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
268194	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
33364	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

5 rows × 4000 columns

In [25]:

```
# Questions 2 tfidf weighted word2vec
df3_q2.head()
```

Out[25]:

	0	1	2	3	4	5	6	7	8	9	...	3990	3991	3992	3993	3994	3995	3996	3997	3998	3999
149483	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

	0	1	2	3	4	5	6	7	8	9	...	3990	3991	3992	3993	3994	3995	3996	3997	3998	3999
146085	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
337094	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
115033	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
190104	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

5 rows × 4000 columns

In [26]:

```
df4_q2.head()
```

Out[26]:

	0	1	2	3	4	5	6	7	8	9	...	3990	3991	3992	3993	3994	3995	3996	3997	3998	3999
303972	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
72206	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
106335	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
268194	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
33364	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

5 rows × 4000 columns

In [27]:

```
print("Number of features in nlp dataframe :", df_tr1.shape[1])
print("Number of features in preprocessed dataframe :", df_tr2.shape[1])
print("Number of features in question1 w2v  dataframe :", df3_q1.shape[1])
print("Number of features in question2 w2v  dataframe :", df3_q2.shape[1])
print("Number of features in final dataframe  :", df_tr1.shape[1]+df_tr2.shape[1]+df3_q1.shape[1]+df3_q
2.shape[1])
```

```
Number of features in nlp dataframe : 17
Number of features in preprocessed dataframe : 12
Number of features in question1 w2v  dataframe : 4000
Number of features in question2 w2v  dataframe : 4000
Number of features in final dataframe  : 8029
```

In [28]:

```
# storing the final features to csv file
if not os.path.isfile('tr_finalfeatures_tfidf.csv'):
    # Assign 'id' attribute astfidf-w2v vector dataframe same as nlp or basic dataframe
    # Please observe above dataframe of basic,nlp and tfidfw2v features, you will find 'id' are all sam
e
    df3_q1['id']=df_tr1['id']
    df3_q2['id']=df_tr1['id']

    # Merge the train basic and nlp feature
    df1  = df_tr1.merge(df_tr2, on='id',how='left')
    print('Total df1 features: {0}'.format(df1.shape))

    # Merge the train tfidf-w2v question1 and question2
    df2  = df3_q1.merge(df3_q2, on='id',how='left')
    print('Total df2 features: {0}'.format(df2.shape))

    # Merge above two dataframe
    result  = df1.merge(df2, on='id',how='left')

    print('Total features: {0}'.format(result.shape))
    result.to_csv('tr_finalfeatures_tfidf.csv')
```

```
Total df1 features: (70000, 28)
Total df2 features: (70000, 8001)
Total features: (70000, 8028)
```

In [29]:

```
# storing the final features to csv file
if not os.path.isfile('ts_finalfeatures_tfidf.csv'):
    # Assign 'id' attribute astfidf-w2v vector dataframe same as nlp or basic dataframe
    # Please observe above dataframe of basic,nlp and tfidfw2v features, you will find 'id' are all sam
e
    df4_q1['id']=df_ts1['id']
    df4_q2['id']=df_ts1['id']

    # Merge the test basic and nlp feature
    df1  = df_ts1.merge(df_ts2, on='id',how='left')
    print('Total df1 features: {0}'.format(df1.shape))

    # Merge the test tfidf-w2v question1 and question2
    df2  = df4_q1.merge(df4_q2, on='id',how='left')
    print('Total df2 features: {0}'.format(df2.shape))

    # Merge above two dataframe
    result  = df1.merge(df2, on='id',how='left')
    print('Total features: {0}'.format(result.shape))
    result.to_csv('ts_finalfeatures_tfidf.csv')
```

```
Total df1 features: (30000, 28)
Total df2 features: (30000, 8001)
Total features: (30000, 8028)
```

In [ ]:

```
```