3.6 Featurizing text data with tfidf weighted word-vectors

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
from tqdm import tqdm
from scipy.sparse import hstack
# exctract word2vec vectors
# https://github.com/explosion/spaCy/issues/1721
# http://landinghub.visualstudio.com/visual-cpp-build-tools
import spacy
from sklearn.model_selection import train_test_split
```

In [2]:

```
# avoid decoding problems
df = pd.read_csv("train.csv")
df=df[:100000]
# 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]:

```
#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')
    dfnlp=dfnlp[:100000]
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')
    dfppro=dfppro[:100000]
else:
    print("download df_fe_without_preprocessing_train.csv from drive or run previous notebook")
```

In [4]:

```
data=pd.concat([df,dfnlp,dfppro],axis=1)
print(data.shape)
#https://stackoverflow.com/questions/14984119/python-pandas-remove-duplicate-columns
data = data.loc[:,~data.columns.duplicated()]
print(data.shape)
```

```
(100000, 44)
(100000, 32)
```

In [5]:

```
y_true = data['is_duplicate']
data.drop(['is_duplicate'], axis=1, inplace=True)
```

In [6]:

```
X_tr,X_test, y_tr, y_test = train_test_split(data, y_true, stratify=y_true, test_size=0.3)
```

In [7]:

```
print("Number of data points in train data :",X_tr.shape)
print("Number of data points in test data :",X_test.shape)
```

```
Number of data points in train data : (70000, 31)
Number of data points in test data : (30000, 31)
```

In [8]:

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

tfidf = TfidfVectorizer(lowercase=False, )
tfidf.fit_transform(questions)

# dict key:word and value:tf-idf score
word2tfidf = dict(zip(tfidf.get_feature_names(), tfidf.idf_))
```

In [9]:

```
# en_vectors_web_lg, which includes over 1 million unique vectors.
nlp = spacy.load('en_core_web_sm')
def vectoriser(source_df,dest_df,source_column,dest_column):
    vecs1 = []
        # https://github.com/noamraph/tqdm
        # tqdm is used to print the progrsource_columness bar
    for qu1 in tqdm(list(source_df[source_column])):
        doc1 = nlp(qu1)
        # 384 is the number of dimensions of vectors
        mean_vec1 = np.zeros([len(doc1), len(doc1[0].vector)])
        for word1 in doc1:
            # word2vec
            vec1 = word1.vector
            # fetch df score
            try:
                idf = word2tfidf[str(word1)]
            except:
                idf = 0
            # compute final vec
            mean_vec1 += vec1 * idf
        mean_vec1 = mean_vec1.mean(axis=0)
        vecs1.append(mean_vec1)
    dest_df[dest_column] = list(vecs1)
```

In [10]:

```
vectoriser(X_tr,X_tr,'question1','q1_feats_m')
vectoriser(X_tr,X_tr,'question2','q2_feats_m')
vectoriser(X_test,X_test,'question1','q1_feats_m')
vectoriser(X_test,X_test,'question2','q2_feats_m')
```

```
100%|████████████████████████████████████████████████████████████| 70000/70000 [22
:37<00:00, 51.55it/s]
100%|████████████████████████████████████████████████████████████| 70000/70000 [23
:10<00:00, 50.36it/s]
100%|████████████████████████████████████████████████████████████| 30000/30000 [09
:50<00:00, 50.76it/s]
100%|████████████████████████████████████████████████████████████| 30000/30000 [09
:56<00:00, 50.31it/s]
```

In [11]:

```
print("Number of data points in train data :",X_tr.shape)
print("Number of data points in test data :",X_test.shape)
```

```
Number of data points in train data : (70000, 33)
Number of data points in test data : (30000, 33)
```

In [12]:

```
#https://stackoverflow.com/questions/40924332/splitting-a-list-in-a-pandas-cell-into-multiple-colu
mns
X_tr1=pd.concat([X_tr,pd.DataFrame(X_tr['q1_feats_m'].values.tolist(),index=X_tr.index,columns=['0_
1','1_1','2_1','3_1','4_1','5_1','6_1','7_1','8_1','9_1','10_1','11_1','12_1','13_1','14_1','15_1',
'16_1','17_1','18_1','19_1','20_1','21_1','22_1','23_1','24_1','25_1','26_1','27_1','28_1','29_1','
30_1','31_1','32_1','33_1','34_1','35_1','36_1','37_1','38_1','39_1','40_1','41_1','42_1','43_1','4
4_1','45_1','46_1','47_1','48_1','49_1','50_1','51_1','52_1','53_1','54_1','55_1','56_1','57_1','58
_1','59_1','60_1','61_1','62_1','63_1','64_1','65_1','66_1','67_1','68_1','69_1','70_1','71_1','72_
1','73_1','74_1','75_1','76_1','77_1','78_1','79_1','80_1','81_1','82_1','83_1','84_1','85_1','86_1
','87_1','88_1','89_1','90_1','91_1','92_1','93_1','94_1','95_1','96_1','97_1','98_1','99_1','100_1'
,'101_1','102_1','103_1','104_1','105_1','106_1','107_1','108_1','109_1','110_1','111_1','112_1','
113_1','114_1','115_1','116_1','117_1','118_1','119_1','120_1','121_1','122_1','123_1','124_1','12
5_1','126_1','127_1','128_1','129_1','130_1','131_1','132_1','133_1','134_1','135_1','136_1','137_
1','138_1','139_1','140_1','141_1','142_1','143_1','144_1','145_1','146_1','147_1','148_1','149_1'
,'150_1','151_1','152_1','153_1','154_1','155_1','156_1','157_1','158_1','159_1','160_1','161_1','
162_1','163_1','164_1','165_1','166_1','167_1','168_1','169_1','170_1','171_1','172_1','173_1','17
4_1','175_1','176_1','177_1','178_1','179_1','180_1','181_1','182_1','183_1','184_1','185_1','186_
1','187_1','188_1','189_1','190_1','191_1','192_1','193_1','194_1','195_1','196_1','197_1','198_1'
,'199_1','200_1','201_1','202_1','203_1','204_1','205_1','206_1','207_1','208_1','209_1','210_1','
211_1','212_1','213_1','214_1','215_1','216_1','217_1','218_1','219_1','220_1','221_1','222_1','22
3_1','224_1','225_1','226_1','227_1','228_1','229_1','230_1','231_1','232_1','233_1','234_1','235_
1','236_1','237_1','238_1','239_1','240_1','241_1','242_1','243_1','244_1','245_1','246_1','247_1'
,'248_1','249_1','250_1','251_1','252_1','253_1','254_1','255_1','256_1','257_1','258_1','259_1','
260_1','261_1','262_1','263_1','264_1','265_1','266_1','267_1','268_1','269_1','270_1','271_1','27
2_1','273_1','274_1','275_1','276_1','277_1','278_1','279_1','280_1','281_1','282_1','283_1','284_
1','285_1','286_1','287_1','288_1','289_1','290_1','291_1','292_1','293_1','294_1','295_1','296_1'
,'297_1','298_1','299_1','300_1','301_1','302_1','303_1','304_1','305_1','306_1','307_1','308_1','
309_1','310_1','311_1','312_1','313_1','314_1','315_1','316_1','317_1','318_1','319_1','320_1','32
1_1','322_1','323_1','324_1','325_1','326_1','327_1','328_1','329_1','330_1','331_1','332_1','333_
1','334_1','335_1','336_1','337_1','338_1','339_1','340_1','341_1','342_1','343_1','344_1','345_1'
,'346_1','347_1','348_1','349_1','350_1','351_1','352_1','353_1','354_1','355_1','356_1','357_1','
358_1','359_1','360_1','361_1','362_1','363_1','364_1','365_1','366_1','367_1','368_1','369_1','37
0_1','371_1','372_1','373_1','374_1','375_1','376_1','377_1','378_1','379_1','380_1','381_1','382_
1','383_1'])], axis=1)
X_tr1=pd.concat([X_tr1,pd.DataFrame(X_tr['q2_feats_m'].values.tolist(),index=X_tr.index,columns=['0
_2','1_2','2_2','3_2','4_2','5_2','6_2','7_2','8_2','9_2','10_2','11_2','12_2','13_2','14_2','15_2'
,'16_2','17_2','18_2','19_2','20_2','21_2','22_2','23_2','24_2','25_2','26_2','27_2','28_2','29_2',
'30_2','31_2','32_2','33_2','34_2','35_2','36_2','37_2','38_2','39_2','40_2','41_2','42_2','43_2','
44_2','45_2','46_2','47_2','48_2','49_2','50_2','51_2','52_2','53_2','54_2','55_2','56_2','57_2','5
8_2','59_2','60_2','61_2','62_2','63_2','64_2','65_2','66_2','67_2','68_2','69_2','70_2','71_2','72
_2','73_2','74_2','75_2','76_2','77_2','78_2','79_2','80_2','81_2','82_2','83_2','84_2','85_2','86_
2','87_2','88_2','89_2','90_2','91_2','92_2','93_2','94_2','95_2','96_2','97_2','98_2','99_2','100_
2','101_2','102_2','103_2','104_2','105_2','106_2','107_2','108_2','109_2','110_2','111_2','112_2'
,'113_2','114_2','115_2','116_2','117_2','118_2','119_2','120_2','121_2','122_2','123_2','124_2','
125_2','126_2','127_2','128_2','129_2','130_2','131_2','132_2','133_2','134_2','135_2','136_2','13
7_2','138_2','139_2','140_2','141_2','142_2','143_2','144_2','145_2','146_2','147_2','148_2','149_
2','150_2','151_2','152_2','153_2','154_2','155_2','156_2','157_2','158_2','159_2','160_2','161_2'
,'162_2','163_2','164_2','165_2','166_2','167_2','168_2','169_2','170_2','171_2','172_2','173_2','
174_2','175_2','176_2','177_2','178_2','179_2','180_2','181_2','182_2','183_2','184_2','185_2','18
6_2','187_2','188_2','189_2','190_2','191_2','192_2','193_2','194_2','195_2','196_2','197_2','198_
2','199_2','200_2','201_2','202_2','203_2','204_2','205_2','206_2','207_2','208_2','209_2','210_2'
,'211_2','212_2','213_2','214_2','215_2','216_2','217_2','218_2','219_2','220_2','221_2','222_2','
223_2','224_2','225_2','226_2','227_2','228_2','229_2','230_2','231_2','232_2','233_2','234_2','23
5_2','236_2','237_2','238_2','239_2','240_2','241_2','242_2','243_2','244_2','245_2','246_2','247_
2','248_2','249_2','250_2','251_2','252_2','253_2','254_2','255_2','256_2','257_2','258_2','259_2'
,'260_2','261_2','262_2','263_2','264_2','265_2','266_2','267_2','268_2','269_2','270_2','271_2','
272_2','273_2','274_2','275_2','276_2','277_2','278_2','279_2','280_2','281_2','282_2','283_2','28
4_2','285_2','286_2','287_2','288_2','289_2','290_2','291_2','292_2','293_2','294_2','295_2','296_
2','297_2','298_2','299_2','300_2','301_2','302_2','303_2','304_2','305_2','306_2','307_2','308_2'
,'309_2','310_2','311_2','312_2','313_2','314_2','315_2','316_2','317_2','318_2','319_2','320_2','
321_2','322_2','323_2','324_2','325_2','326_2','327_2','328_2','329_2','330_2','331_2','332_2','33
3_2','334_2','335_2','336_2','337_2','338_2','339_2','340_2','341_2','342_2','343_2','344_2','345_
2','346_2','347_2','348_2','349_2','350_2','351_2','352_2','353_2','354_2','355_2','356_2','357_2'
```

```
,'358_2','359_2','360_2','361_2','362_2','363_2','364_2','365_2','366_2','367_2','368_2','369_2','
370_2','371_2','372_2','373_2','374_2','375_2','376_2','377_2','378_2','379_2','380_2','381_2','38
2_2','383_2'])], axis=1)
X_test1=pd.concat([X_test,pd.DataFrame(X_test['q1_feats_m'].values.tolist(),index=X_test.index,colu
mns=['0_1','1_1','2_1','3_1','4_1','5_1','6_1','7_1','8_1','9_1','10_1','11_1','12_1','13_1','14_1'
,'15_1','16_1','17_1','18_1','19_1','20_1','21_1','22_1','23_1','24_1','25_1','26_1','27_1','28_1',
'29_1','30_1','31_1','32_1','33_1','34_1','35_1','36_1','37_1','38_1','39_1','40_1','41_1','42_1','
43_1','44_1','45_1','46_1','47_1','48_1','49_1','50_1','51_1','52_1','53_1','54_1','55_1','56_1','5
7_1','58_1','59_1','60_1','61_1','62_1','63_1','64_1','65_1','66_1','67_1','68_1','69_1','70_1','71
_1','72_1','73_1','74_1','75_1','76_1','77_1','78_1','79_1','80_1','81_1','82_1','83_1','84_1','85_
1','86_1','87_1','88_1','89_1','90_1','91_1','92_1','93_1','94_1','95_1','96_1','97_1','98_1','99_1
','100_1','101_1','102_1','103_1','104_1','105_1','106_1','107_1','108_1','109_1','110_1','111_1',
'112_1','113_1','114_1','115_1','116_1','117_1','118_1','119_1','120_1','121_1','122_1','123_1','1
24_1','125_1','126_1','127_1','128_1','129_1','130_1','131_1','132_1','133_1','134_1','135_1','136
_1','137_1','138_1','139_1','140_1','141_1','142_1','143_1','144_1','145_1','146_1','147_1','148_1'
,'149_1','150_1','151_1','152_1','153_1','154_1','155_1','156_1','157_1','158_1','159_1','160_1','
161_1','162_1','163_1','164_1','165_1','166_1','167_1','168_1','169_1','170_1','171_1','172_1','17
3_1','174_1','175_1','176_1','177_1','178_1','179_1','180_1','181_1','182_1','183_1','184_1','185_
1','186_1','187_1','188_1','189_1','190_1','191_1','192_1','193_1','194_1','195_1','196_1','197_1'
,'198_1','199_1','200_1','201_1','202_1','203_1','204_1','205_1','206_1','207_1','208_1','209_1','
210_1','211_1','212_1','213_1','214_1','215_1','216_1','217_1','218_1','219_1','220_1','221_1','22
2_1','223_1','224_1','225_1','226_1','227_1','228_1','229_1','230_1','231_1','232_1','233_1','234_
1','235_1','236_1','237_1','238_1','239_1','240_1','241_1','242_1','243_1','244_1','245_1','246_1'
,'247_1','248_1','249_1','250_1','251_1','252_1','253_1','254_1','255_1','256_1','257_1','258_1','
259_1','260_1','261_1','262_1','263_1','264_1','265_1','266_1','267_1','268_1','269_1','270_1','27
1_1','272_1','273_1','274_1','275_1','276_1','277_1','278_1','279_1','280_1','281_1','282_1','283_
1','284_1','285_1','286_1','287_1','288_1','289_1','290_1','291_1','292_1','293_1','294_1','295_1'
,'296_1','297_1','298_1','299_1','300_1','301_1','302_1','303_1','304_1','305_1','306_1','307_1','
308_1','309_1','310_1','311_1','312_1','313_1','314_1','315_1','316_1','317_1','318_1','319_1','32
0_1','321_1','322_1','323_1','324_1','325_1','326_1','327_1','328_1','329_1','330_1','331_1','332_
1','333_1','334_1','335_1','336_1','337_1','338_1','339_1','340_1','341_1','342_1','343_1','344_1'
,'345_1','346_1','347_1','348_1','349_1','350_1','351_1','352_1','353_1','354_1','355_1','356_1','
357_1','358_1','359_1','360_1','361_1','362_1','363_1','364_1','365_1','366_1','367_1','368_1','36
9_1','370_1','371_1','372_1','373_1','374_1','375_1','376_1','377_1','378_1','379_1','380_1','381_
1','382_1','383_1'])], axis=1)
X_test1=pd.concat([X_test1,pd.DataFrame(X_test['q1_feats_m'].values.tolist(),index=X_test.index,col
umns=['0_2','1_2','2_2','3_2','4_2','5_2','6_2','7_2','8_2','9_2','10_2','11_2','12_2','13_2','14_2
','15_2','16_2','17_2','18_2','19_2','20_2','21_2','22_2','23_2','24_2','25_2','26_2','27_2','28_2'
,'29_2','30_2','31_2','32_2','33_2','34_2','35_2','36_2','37_2','38_2','39_2','40_2','41_2','42_2',
'43_2','44_2','45_2','46_2','47_2','48_2','49_2','50_2','51_2','52_2','53_2','54_2','55_2','56_2','
57_2','58_2','59_2','60_2','61_2','62_2','63_2','64_2','65_2','66_2','67_2','68_2','69_2','70_2','7
1_2','72_2','73_2','74_2','75_2','76_2','77_2','78_2','79_2','80_2','81_2','82_2','83_2','84_2','85
_2','86_2','87_2','88_2','89_2','90_2','91_2','92_2','93_2','94_2','95_2','96_2','97_2','98_2','99_
2','100_2','101_2','102_2','103_2','104_2','105_2','106_2','107_2','108_2','109_2','110_2','111_2'
,'112_2','113_2','114_2','115_2','116_2','117_2','118_2','119_2','120_2','121_2','122_2','123_2','
124_2','125_2','126_2','127_2','128_2','129_2','130_2','131_2','132_2','133_2','134_2','135_2','13
6_2','137_2','138_2','139_2','140_2','141_2','142_2','143_2','144_2','145_2','146_2','147_2','148_
2','149_2','150_2','151_2','152_2','153_2','154_2','155_2','156_2','157_2','158_2','159_2','160_2'
,'161_2','162_2','163_2','164_2','165_2','166_2','167_2','168_2','169_2','170_2','171_2','172_2','
173_2','174_2','175_2','176_2','177_2','178_2','179_2','180_2','181_2','182_2','183_2','184_2','18
5_2','186_2','187_2','188_2','189_2','190_2','191_2','192_2','193_2','194_2','195_2','196_2','197_
2','198_2','199_2','200_2','201_2','202_2','203_2','204_2','205_2','206_2','207_2','208_2','209_2'
,'210_2','211_2','212_2','213_2','214_2','215_2','216_2','217_2','218_2','219_2','220_2','221_2','
222_2','223_2','224_2','225_2','226_2','227_2','228_2','229_2','230_2','231_2','232_2','233_2','23
4_2','235_2','236_2','237_2','238_2','239_2','240_2','241_2','242_2','243_2','244_2','245_2','246_
2','247_2','248_2','249_2','250_2','251_2','252_2','253_2','254_2','255_2','256_2','257_2','258_2'
,'259_2','260_2','261_2','262_2','263_2','264_2','265_2','266_2','267_2','268_2','269_2','270_2','
271_2','272_2','273_2','274_2','275_2','276_2','277_2','278_2','279_2','280_2','281_2','282_2','28
3_2','284_2','285_2','286_2','287_2','288_2','289_2','290_2','291_2','292_2','293_2','294_2','295_
2','296_2','297_2','298_2','299_2','300_2','301_2','302_2','303_2','304_2','305_2','306_2','307_2'
,'308_2','309_2','310_2','311_2','312_2','313_2','314_2','315_2','316_2','317_2','318_2','319_2','
320_2','321_2','322_2','323_2','324_2','325_2','326_2','327_2','328_2','329_2','330_2','331_2','33
2_2','333_2','334_2','335_2','336_2','337_2','338_2','339_2','340_2','341_2','342_2','343_2','344_
2','345_2','346_2','347_2','348_2','349_2','350_2','351_2','352_2','353_2','354_2','355_2','356_2'
,'357_2','358_2','359_2','360_2','361_2','362_2','363_2','364_2','365_2','366_2','367_2','368_2','
369_2','370_2','371_2','372_2','373_2','374_2','375_2','376_2','377_2','378_2','379_2','380_2','38
1_2','382_2','383_2'])], axis=1)
```

In [13]:

```
print("Number of data points in train data :",X_tr.shape)
print("Number of data points in test data :",X_test.shape)
print("Number of data points in train data :",X_tr1.shape)
print("Number of data points in test data :",X_test1.shape)
```

```
Number of data points in train data : (70000, 33)
```

```
Number of data points in test data : (30000, 33)
Number of data points in train data : (70000, 801)
Number of data points in test data : (30000, 801)
```

In [14]:

```
X_tr.drop(['q1_feats_m','q2_feats_m'], axis=1,inplace=True)
X_test.drop(['q1_feats_m','q2_feats_m'], axis=1,inplace=True)
X_tr1.drop(['qid1','qid2','question1','question2','q1_feats_m','q2_feats_m'], axis=1,inplace=True)
X_test1.drop(['qid1','qid2','question1','question2','q1_feats_m','q2_feats_m'],
axis=1,inplace=True)
```

In [15]:

```
print("Number of data points in train data :",X_tr.shape)
print("Number of data points in test data :",X_test.shape)
print("Number of data points in train data :",X_tr1.shape)
print("Number of data points in test data :",X_test1.shape)
```

```
Number of data points in train data : (70000, 31)
Number of data points in test data : (30000, 31)
Number of data points in train data : (70000, 795)
Number of data points in test data : (30000, 795)
```

In [16]:

```
from sqlalchemy import create_engine
engine = create_engine('sqlite:///w2v_data.db')

#X_tr_q1  = X_tr_q1.merge(X_tr_q2, on='id',how='left')
#w2v_tr  = df1_tr.merge(X_tr_q1, on='id',how='left')
print(X_tr1.shape)
X_tr1.to_sql('X_tr', engine, if_exists='replace')
y_tr.to_sql('y_tr', engine, if_exists='replace')

#X_test_q1  = X_test_q1.merge(X_test_q2, on='id',how='left')
#w2v_test  = df1_test.merge(X_test_q1, on='id',how='left')
print(X_test1.shape)
X_test1.to_sql('X_test', engine, if_exists='replace')
y_test.to_sql('y_test', engine, if_exists='replace')
```

```
(70000, 795)
(30000, 795)
```

In [96]:

```
print("Number of data points in train data :",X_tr.shape)
print("Number of data points in test data :",X_test.shape)
```

```
Number of data points in train data : (70000, 31)
Number of data points in test data : (30000, 31)
```

In [97]:

```
#https://stackoverflow.com/questions/45961747/append-tfidf-to-pandas-dataframe
from sklearn.feature_extraction.text import TfidfVectorizer

tfidf = TfidfVectorizer(min_df=10)
tfidf_ques1_tr= tfidf.fit_transform(X_tr['question1'])
tfidf_ques1_test= tfidf.transform(X_test['question1'])

X_q1_tr = pd.DataFrame(tfidf_ques1_tr.toarray(), columns=tfidf.get_feature_names(), index=
X_tr.index)
X_q1_test = pd.DataFrame(tfidf_ques1_test.toarray(), columns=tfidf.get_feature_names(), index=
X_test.index)
#X_tr['q1_feats_m'] = list(tfidf_ques1_tr.toarray())
#X_test['q1_feats_m'] = list(tfidf_ques1_test.toarray())
#X_tr['q2_feats_m'] = list(tfidf_ques2_tr.toarray())
#X_test['q2_feats_m'] = list(tfidf_ques2_test.toarray())
```

- After we find TF-IDF scores, we convert each question to a weighted average of word2vec vectors by these scores.
- here we use a pre-trained GLOVE model which comes free with "Spacy". https://spacy.io/usage/vectors-similarity
- It is trained on Wikipedia and therefore, it is stronger in terms of word semantics.

In [99]:

```
#X_q1_tr = pd.DataFrame(tfidf_ques1_tr.toarray(), index= X_tr.index)

tfidf = TfidfVectorizer(min_df=10)
tfidf_ques2_tr= tfidf.fit_transform(X_tr['question2'])
tfidf_ques2_test= tfidf.transform(X_test['question2'])

X_q2_tr = pd.DataFrame(tfidf_ques2_tr.toarray(), columns=tfidf.get_feature_names(), index=
X_tr.index)
X_q2_test = pd.DataFrame(tfidf_ques2_test.toarray(), columns=tfidf.get_feature_names(), index=
X_test.index)
```

In [101]:

```
print("Number of data points in train data :",X_tr.shape)
print("Number of data points in test data :",X_test.shape)
```

```
Number of data points in train data : (70000, 31)
Number of data points in test data : (30000, 31)
```

In [102]:

```
X_tr.drop(['qid1','qid2','question1','question2'],axis=1,inplace=True)
X_test.drop(['qid1','qid2','question1','question2'],axis=1,inplace=True)
```

In [106]:

```
X_tr=pd.concat([X_tr,X_q1_tr,X_q2_tr], axis=1)
X_test=pd.concat([X_test,X_q1_test,X_q2_test], axis=1)
```

In [108]:

```
print("Number of data points in train data :",X_tr.shape)
print("Number of data points in test data :",X_test.shape)
```

```
Number of data points in train data : (70000, 10431)
Number of data points in test data : (30000, 10431)
```

In [109]:

```
#X_q1_tr  = X_q1_tr.merge(X_q2_tr, on='id',how='left')
#tfidf_tr  = df2_tr.merge(X_q1_tr, on='id',how='left')
print(X_tr.shape)
X_tr.to_csv('X_tr.csv',index='id')
#X_q1_test  = X_q1_test.merge(X_q2_test, on='id',how='left')
#tfidf_test  = df2_test.merge(X_q1_test, on='id',how='left')
print(X_test.shape)
X_test.to_csv('X_test.csv',index='id')
```

```
(70000, 10431)
(30000, 10431)
```