

3.Q_Mean_W2V

February 27, 2019

3.6 Featurizing text data with tfidf weighted word-vectors

```
In [3]: df=pd.read_csv('final_features.csv')
```

```
In [4]: df.shape
```

```
Out[4]: (404290, 797)
```

```
In [2]: 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
```

```
# extract word2vec vectors
# https://github.com/explosion/spaCy/issues/1721
# http://landinghub.visualstudio.com/visual-cpp-build-tools
import spacy
```

```
In [3]: # 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 [4]: df.head()
```

```
Out[4]:
```

	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 23^{24} is...	0
4	4	9	10	Which one dissolve in water quikly sugar, salt...	Which fish would survive in salt water?	0

```
In [5]: from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.feature_extraction.text import CountVectorizer
# merge texts
questions = list(df['question1']) + list(df['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_))
```

- 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 [7]: # en_vectors_web_lg, which includes over 1 million unique vectors.
nlp = spacy.load('en_core_web_sm')

vecs1 = []
# https://github.com/noamraph/tqdm
# tqdm is used to print the progress bar
for qu1 in tqdm(list(df['question1'])):
    doc1 = nlp(qu1)
    # 384 is the number of dimensions of vectors
    mean_vec1 = np.zeros([len(doc1), 384])
    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)
df['q1_feats_m'] = list(vecs1)

```

100%|| 404290/404290 [1:24:04<00:00, 80.14it/s]

```

In [8]: vecs2 = []
        for qu2 in tqdm(list(df['question2'])):
            doc2 = nlp(qu2)
            mean_vec2 = np.zeros([len(doc2), 384])
            for word2 in doc2:
                # word2vec
                vec2 = word2.vector
                # fetch df score
                try:
                    idf = word2tfidf[str(word2)]
                except:
                    #print word
                    idf = 0
                # compute final vec
                mean_vec2 += vec2 * idf
            mean_vec2 = mean_vec2.mean(axis=0)
            vecs2.append(mean_vec2)
df['q2_feats_m'] = list(vecs2)

```

100%|| 404290/404290 [1:26:23<00:00, 77.99it/s]

```

In [9]: #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 [10]: df1 = dfnlp.drop(['qid1','qid2','question1','question2'],axis=1)
df2 = dfppro.drop(['qid1','qid2','question1','question2','is_duplicate'],axis=1)
df3 = df.drop(['qid1','qid2','question1','question2','is_duplicate'],axis=1)
df3_q1 = pd.DataFrame(df3.q1_feats_m.values.tolist(), index= df3.index)
df3_q2 = pd.DataFrame(df3.q2_feats_m.values.tolist(), index= df3.index)
```

```
In [11]: # dataframe of nlp features
df1.head()
```

```
Out[11]:
```

	id	is_duplicate	cwc_min	cwc_max	csc_min	csc_max	ctc_min	\
0	0	0	0.999980	0.833319	0.999983	0.999983	0.916659	
1	1	0	0.799984	0.399996	0.749981	0.599988	0.699993	
2	2	0	0.399992	0.333328	0.399992	0.249997	0.399996	
3	3	0	0.000000	0.000000	0.000000	0.000000	0.000000	
4	4	0	0.399992	0.199998	0.999950	0.666644	0.571420	

	ctc_max	last_word_eq	first_word_eq	abs_len_diff	mean_len	\
0	0.785709	0.0	1.0	2.0	13.0	
1	0.466664	0.0	1.0	5.0	12.5	
2	0.285712	0.0	1.0	4.0	12.0	
3	0.000000	0.0	0.0	2.0	12.0	
4	0.307690	0.0	1.0	6.0	10.0	

	token_set_ratio	token_sort_ratio	fuzz_ratio	fuzz_partial_ratio	\
0	100	93	93	100	
1	86	63	66	75	
2	63	63	43	47	
3	28	24	9	14	
4	67	47	35	56	

	longest_substr_ratio
0	0.982759
1	0.596154
2	0.166667
3	0.039216
4	0.175000

```
In [12]: # data before preprocessing
df2.head()
```

```
Out[12]:
```

	id	freq_qid1	freq_qid2	q1len	q2len	q1_n_words	q2_n_words	\
0	0	1	1	66	57	14	12	
1	1	4	1	51	88	8	13	
2	2	1	1	73	59	14	10	
3	3	1	1	50	65	11	9	
4	4	3	1	76	39	13	7	

	word_Common	word_Total	word_share	freq_q1+q2	freq_q1-q2
0	10.0	23.0	0.434783	2	0

1	4.0	20.0	0.200000	5	3
2	4.0	24.0	0.166667	2	0
3	0.0	19.0	0.000000	2	0
4	2.0	20.0	0.100000	4	2

```
In [13]: # Questions 1 tfidf weighted word2vec
df3_q1.head()
```

```
Out[13]:
```

	0	1	2	3	4	5	\
0	121.929927	100.083906	72.497900	115.641795	-48.370865	34.619070	
1	-78.070935	54.843787	82.738495	98.191855	-51.234840	55.013509	
2	-5.355015	73.671810	14.376365	104.130241	1.433537	35.229116	
3	5.778359	-34.712038	48.999631	59.699204	40.661263	-41.658731	
4	51.138220	38.587312	123.639488	53.333041	-47.062739	37.356212	

	6	7	8	9	...	374	\
0	-172.057790	-92.502626	113.223311	50.562456	...	12.397640	
1	-39.140733	-82.692374	45.161483	-9.556298	...	-21.987079	
2	-148.519385	-97.124595	41.972195	50.948731	...	3.027700	
3	-36.808594	24.170655	0.235601	-29.407290	...	13.100007	
4	-298.722753	-106.421119	106.248914	65.880707	...	13.906532	

	375	376	377	378	379	380	381	\
0	40.909521	8.150260	-15.170692	18.007711	6.166997	-30.124162	3.700903	
1	-12.389279	20.667980	2.202720	-17.142453	-5.880972	-10.123963	-4.890665	
2	14.025767	-2.960312	-3.206544	4.355141	2.936152	-20.199555	9.816351	
3	1.405671	-1.891076	-7.882638	18.000561	12.106918	-10.507835	5.243834	
4	43.461721	11.519207	-22.468284	45.431128	8.161224	-35.373910	7.728865	

	382	383
0	-1.757692	-1.818057
1	-13.018393	-5.219309
2	11.894366	-8.798819
3	10.158340	5.886351
4	9.592849	5.447336

[5 rows x 384 columns]

```
In [14]: # Questions 2 tfidf weighted word2vec
df3_q2.head()
```

```
Out[14]:
```

	0	1	2	3	4	5	\
0	125.983301	95.636484	42.114717	95.449986	-37.386301	39.400084	
1	-106.871899	80.290340	79.066300	59.302100	-42.175332	117.616657	
2	7.072875	15.513378	1.846914	85.937583	-33.808811	94.702337	
3	39.421539	44.136990	-24.010927	85.265864	-0.339028	-9.323141	
4	31.950109	62.854102	1.778147	36.218763	-45.130861	66.674880	

	6	7	8	9	...	374	\
--	---	---	---	---	-----	-----	---

0	-148.116068	-87.851481	110.371972	62.272816	...	16.165591
1	-144.364242	-127.131506	22.962531	25.397579	...	-4.901130
2	-122.256856	-114.009530	53.922293	60.131814	...	8.359966
3	-60.499653	-37.044767	49.407847	-23.350149	...	3.311411
4	-106.342344	-22.901031	59.835921	62.663957	...	-2.403871

	375	376	377	378	379	380 \
0	33.030668	7.019997	-14.793958	15.437512	8.199657	-25.070835
1	-4.565391	41.520751	-0.727564	-16.413779	-7.373777	2.638877
2	-2.165985	10.936580	-16.531660	14.681230	15.633759	-1.210901
3	3.788878	13.398598	-6.592596	6.437365	5.993291	2.732392
4	11.991207	8.088480	-15.090196	8.375168	1.727225	-6.601130

	381	382	383
0	1.571618	1.603736	0.305646
1	-7.403460	2.703069	0.408038
2	14.183826	11.703135	10.148075
3	-3.727647	5.614115	6.023694
4	11.317404	11.544597	2.478685

[5 rows x 384 columns]

```
In [15]: print("Number of features in nlp dataframe :", df1.shape[1])
         print("Number of features in preprocessed dataframe :", df2.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 :", df1.shape[1]+df2.shape[1]+df3_q1.sh
```

```
Number of features in nlp dataframe : 17
Number of features in preprocessed dataframe : 12
Number of features in question1 w2v dataframe : 384
Number of features in question2 w2v dataframe : 384
Number of features in final dataframe : 797
```

```
In [16]: # storing the final features to csv file
         if not os.path.isfile('final_features.csv'):
             df3_q1['id']=df1['id']
             df3_q2['id']=df1['id']
             df1 = df1.merge(df2, on='id',how='left')
             df2 = df3_q1.merge(df3_q2, on='id',how='left')
             result = df1.merge(df2, on='id',how='left')
             result.to_csv('final_features.csv')
```

```
In [1]: df1.head()
```

NameError

Traceback (most recent call last)

```
<ipython-input-1-eda2b5cc1a42> in <module>  
----> 1 df1.head()
```

NameError: name 'df1' is not defined

In []: