3.Q_Mean_W2V

April 18, 2019

3.6 Featurizing text data with tfidf weighted word-vectors

```
In [42]: 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

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

In [43]: # 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 [44]: df.head()
```

```
Out[44]:    id  qid1  qid2                                         question1  \
        0   0     1     2  What is the step by step guide to invest in sh...
        1   1     3     4  What is the story of Kohinoor (Koh-i-Noor) Dia...
        2   2     5     6  How can I increase the speed of my internet co...
        3   3     7     8  Why am I mentally very lonely? How can I solve...
        4   4     9    10  Which one dissolve in water quikly sugar, salt...

                                                   question2  is_duplicate
        0  What is the step by step guide to invest in sh...             0
        1  What would happen if the Indian government sto...             0
        2  How can Internet speed be increased by hacking...             0
        3  Find the remainder when [math]23^{24}[/math] i...             0
        4            Which fish would survive in salt water?             0
```

```
In [45]: # 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_))
```

```
In [ ]: word2tfidf
```

- 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 [23]: # en_vectors_web_lg, which includes over 1 million unique vectors.
         nlp = spacy.load(r"C:\Users\Byron\Applications\PythonMaster\Lib\site-packages\en_core
```

```
In [32]: 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), 300])
             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 [56:22<00:00, 119.54it/s]
```

```
In [33]: vecs2 = []
         for qu2 in tqdm(list(df['question2'])):
             doc2 = nlp(qu2)
             mean_vec2 = np.zeros([len(doc2), 300])
             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 [58:32<00:00, 115.11it/s]
```

```
In [34]: #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 
```

```
In [35]: 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 [36]: # dataframe of nlp features
         df1.head()
```

```
Out[36]:    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              1.000000
         1              0.607843
         2              0.169492
         3              0.040000
         4              0.179487
```

```
In [37]: # data before preprocessing
         df2.head()
```

```
Out[37]:    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 [38]: # Questions 1 tfidf weighted word2vec
         df3_q1.head()
```

```
Out[38]:            0          1          2         3          4          5          6  \
         0  -5.856872  17.449559   4.862720  7.971019  20.345586  -5.514759  -4.077800
         1   9.356103  13.098566  18.945098 -2.079594 -15.703841  -2.173409   8.969065
         2   0.909520  16.050299  -8.126856 -4.848289  -2.806190   9.752280   4.349992
         3  -4.950745  17.098874 -15.474965  1.044680  -2.392017  -0.051889   2.650595
         4 -11.520302  19.769948  -4.510997 -6.548994 -20.835286  33.663909 -30.390504

                    7          8           9     ...            290       291  \
         0  -2.820742   8.029026  146.599092     ...     -17.370964  5.393082
         1 -20.458267 -20.674299   13.760798     ...      25.948247  0.603713
         2  -5.120332   6.785252  106.342974     ...     -20.942061  2.398984
         3  -8.451192   2.584123  116.184408     ...      -2.551312 -4.971480
         4   0.826553 -19.571472   84.458577     ...      -8.331733 -4.866335

                  292        293        294        295        296        297        298  \
         0   0.384676  -8.362788  -1.880290 -10.799672 -12.999799   3.225858   1.256145
         1 -10.516349   6.040723  30.476707   3.976890 -28.254610  12.613432  -7.770673
         2   8.663028  -0.654124  16.220601  -2.719094  10.485332  -1.103132  -7.290877
         3  -0.478381  -1.930166   9.336016   2.574459   4.803863  -1.182989  -2.962115
         4  18.828458 -40.357679 -10.336167  15.294630  -0.989347  -9.072091  -8.194567

                  299
         0  16.807275
         1  31.456654
         2  19.314250
         3   3.225704
         4  23.847560

         [5 rows x 300 columns]
```

```
In [39]: # Questions 2 tfidf weighted word2vec
         df3_q2.head()
```

```
Out[39]:            0          1          2         3          4         5          6  \
         0   0.398579  13.991607  -0.504564  9.254431  13.906436 -4.777694  -5.274421
         1   4.649688   9.974928  20.330103 -0.440372 -18.128566 -1.984671   4.906458
         2 -17.305105  17.355614  -9.135664 -6.038550  -1.831651  4.547895  17.935764
         3   3.897911   2.545857  -2.053792  3.385450   3.424216 -2.282545 -11.763825
         4  -5.391206   1.767221   1.810128 -4.097073  -3.623262  8.417368 -25.246265

                    7          8           9     ...            290       291  \
         0  -0.201208   4.940558  134.735950     ...     -17.810438  7.231024
         1 -27.797837 -21.262646   96.965297     ...      23.015827  3.435464
         2  -4.799029   3.100311   99.380095     ...     -24.310109 -1.216773
         3   6.692485   5.797674   94.978085     ...      -5.435584  1.672591
         4   7.473430  -2.789541   89.594627     ...     -10.407441 -8.444207

                  292        293        294        295        296        297  \
         0   1.531186  -7.528823   0.473802 -11.864658 -11.293788   1.866265
         1  -5.169600   7.102491  34.516881   6.177686 -27.770856  12.926435
         2  11.909693   9.591573  11.846737   1.397859   6.454157  -0.271460
         3  -0.863278  -2.906553  -3.466688  -3.867892  -4.249463 -12.551012
         4 -14.450059 -12.709382  -4.449050  12.563987 -11.721362 -16.459300

                  298        299
         0   3.616046  11.971096
         1  -4.564559  33.919834
         2 -12.500337  27.634567
         3   4.494087  -6.223341
         4   3.626297  -9.790615

         [5 rows x 300 columns]
```

```
In [40]: 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.sha
```

```
Number of features in nlp dataframe : 17
Number of features in preprocessed dataframe : 12
Number of features in question1 w2v  dataframe : 300
Number of features in question2 w2v  dataframe : 300
Number of features in final dataframe  : 629
```

```
In [41]: # 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')
```