generate train test data for tfidf and tfidf averaged w2v

October 30, 2019

1.1 Featurizing text data with tfidf weighted word-vectors

[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

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

import pandas as pd
import matplotlib.pyplot as plt
import re
import time
import warnings
import sqlite3
from sqlalchemy import create_engine # database connection
import csv
import os
warnings.filterwarnings("ignore")
import datetime as dt
import numpy as np
from nltk.corpus import stopwords
```

```
from sklearn.decomposition import TruncatedSVD
from sklearn.preprocessing import normalize
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.manifold import TSNE
import seaborn as sns
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import confusion_matrix
from sklearn.metrics.classification import accuracy_score, log_loss
from sklearn.feature_extraction.text import TfidfVectorizer
from collections import Counter
from scipy.sparse import hstack
from sklearn.multiclass import OneVsRestClassifier
from sklearn.svm import SVC
from sklearn.model_selection import StratifiedKFold
from collections import Counter, defaultdict
from sklearn.calibration import CalibratedClassifierCV
from sklearn.naive_bayes import MultinomialNB
from sklearn.naive_bayes import GaussianNB
from sklearn.model_selection import train_test_split
from sklearn.model_selection import GridSearchCV
import math
from sklearn.metrics import normalized_mutual_info_score
from sklearn.ensemble import RandomForestClassifier



from sklearn.model_selection import cross_val_score
from sklearn.linear_model import SGDClassifier
from mlxtend.classifier import StackingClassifier

from sklearn import model_selection
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import precision_recall_curve, auc, roc_curve
```

[3]:
```
# avoid decoding problems
df = pd.read_csv("train.csv")
df = df.sample(200000,random_state=42)

# 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))
```

[4]:
```
X = df
print(X.columns)
y = X['is_duplicate']
X = X.drop('is_duplicate',axis=1)
print(X.columns)
```

```
Index(['id', 'qid1', 'qid2', 'question1', 'question2', 'is_duplicate'],
dtype='object')
Index(['id', 'qid1', 'qid2', 'question1', 'question2'], dtype='object')
```

[5]:
```
X_train,X_test, y_train, y_test = train_test_split(X, y, stratify=y,␣
↪test_size=0.3)
```

[6]:
```
y_train.to_csv('final_features_tfidf_train_label.csv')
y_test.to_csv('final_features_tfidf_test_label.csv')
```

[7]:
```
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.feature_extraction.text import CountVectorizer
# merge texts
questions = list(X_train['question1']) + list(X_train['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_))
```

[8]:
```
# 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(X_train['question1'])):
    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)
```

```
X_train['q1_feats_m'] = list(vecs1)
```

```
100%||
140000/140000 [14:45<00:00, 158.16it/s]
```

[9]:
```
# 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(X_train['question2'])):
    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)
X_train['q2_feats_m'] = list(vecs1)
```

```
100%||
140000/140000 [14:43<00:00, 158.48it/s]
```

[10]:
```
# 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(X_test['question1'])):
    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)
X_test['q1_feats_m'] = list(vecs1)
```

```
100%|
| 60000/60000 [06:13<00:00, 160.62it/s]
```

[11]:
```
# 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(X_test['question2'])):
    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)
X_test['q2_feats_m'] = list(vecs1)
```

```
100%|
| 60000/60000 [06:16<00:00, 159.28it/s]
```

[12]:
```
#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")
```

[13]:
```
df1 = dfnlp.drop(['qid1','qid2','question1','question2'],axis=1)
df2 = dfppro.drop(['qid1','qid2','question1','question2','is_duplicate'],axis=1)
df3 = X_test.drop(['qid1','qid2','question1','question2'],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)

df3_q1.index.name = 'id'
df3_q2.index.name = 'id'
df_questions_merged  = df3_q1.merge(df3_q2,on = 'id')
final_df_test = df_questions_merged.merge(df1, on ='id')
final_df_test = final_df_test.merge(df2, on ='id')
if not os.path.isfile('final_features_w2v_test.csv'):
    final_df_test.to_csv('final_features_w2v_test.csv')
```

[14]:
```
df1 = dfnlp.drop(['qid1','qid2','question1','question2'],axis=1)
df2 = dfppro.drop(['qid1','qid2','question1','question2','is_duplicate'],axis=1)
df3 = X_train.drop(['qid1','qid2','question1','question2'],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)

df3_q1.index.name = 'id'
df3_q2.index.name = 'id'
df_questions_merged  = df3_q1.merge(df3_q2,on = 'id')
final_df_train = df_questions_merged.merge(df1, on ='id')
final_df_train = final_df_train.merge(df2, on ='id')
if not os.path.isfile('final_features_w2v_train.csv'):
    final_df_train.to_csv('final_features_w2v_train.csv')
```

1.2 Featurizing text data with tfidf

[15]:
```
from sklearn.feature_extraction.text import TfidfVectorizer
# merge texts
questions = list(X_train['question1']) + list(X_train['question2'])

tfidf = TfidfVectorizer(lowercase=False, max_features=1000)
tfidf.fit(questions)
X_train['q1_feats_m'] = list(tfidf.transform(X_train['question1']).toarray())
X_train['q2_feats_m'] = list(tfidf.transform(X_train['question2']).toarray())
X_test['q1_feats_m'] = list(tfidf.transform(X_test['question1']).toarray())
X_test['q2_feats_m'] = list(tfidf.transform(X_test['question2']).toarray())
```

[16]:
```
#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")
```

[17]:
```
df1 = dfnlp.drop(['qid1','qid2','question1','question2'],axis=1)
df2 = dfppro.drop(['qid1','qid2','question1','question2','is_duplicate'],axis=1)
df3 = X_test.drop(['qid1','qid2','question1','question2'],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)

df3_q1.index.name = 'id'
df3_q2.index.name = 'id'
df_questions_merged  = df3_q1.merge(df3_q2,on = 'id')
final_df_test = df_questions_merged.merge(df1, on ='id')
final_df_test = final_df_test.merge(df2, on ='id')
if not os.path.isfile('final_features_tfidf_test.csv'):
    final_df_test.to_csv('final_features_tfidf_test.csv')
```

[18]:
```
df1 = dfnlp.drop(['qid1','qid2','question1','question2'],axis=1)
df2 = dfppro.drop(['qid1','qid2','question1','question2','is_duplicate'],axis=1)
df3 = X_train.drop(['qid1','qid2','question1','question2'],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)

df3_q1.index.name = 'id'
df3_q2.index.name = 'id'
df_questions_merged  = df3_q1.merge(df3_q2,on = 'id')
final_df_train = df_questions_merged.merge(df1, on ='id')
final_df_train = final_df_train.merge(df2, on ='id')
if not os.path.isfile('final_features_tfidf_train.csv'):
    final_df_train.to_csv('final_features_tfidf_train.csv')
```