notebook

April 18, 2018

1 Domain classification

1.0.1 Emanuele Alessi 1486470

```
In [1]: import pickle
        import numpy as np
        import os
        import tqdm
        from data_preprocessing import preprocess_text
        import seaborn as sn
        import pandas as pd
        import matplotlib.pyplot as plt
```

```
In [2]: embedding = pickle.load(open('vectors.pkl', 'rb'))
        emb = {}
        row = 0
        for word in open('tmp/metadata.tsv', 'r'):
            emb[word.replace('\n','')] = row
            row += 1
```

```
In [3]: def read_data(directory):
            dataset = np.array([])
            stopwords = set([w.rstrip('\r\n') for w in open('stopwords.txt')])
            for domain in os.listdir(directory):
                print(domain)
                files = os.listdir(os.path.join(directory, domain))
                for i in tqdm.tqdm(range(len(files))):
                    if files[i].endswith(".txt"):
                        with open(os.path.join(directory, domain, files[i]), encoding='utf8') a
                            data = []
                            for line in file.readlines():
                                split = preprocess_text(line, stopwords)  # split is a list of
                                for sentence in split:
                                    if sentence:
                                        data += [sentence]
                            val = embeddings_mean(embedding, emb, data, domain)
                            if val is not None:
                                if dataset.size == 0:
                                    dataset = np.append(dataset, val)
                                else:
                                    dataset = np.vstack((dataset, val))
            return dataset

        def embeddings_mean(embedding, emb, data, domain=None):
            word_count = 0
            s = 0
            for sentence in data:
                for word in sentence:
                    if word in emb:
                        s += embedding[emb[word]]
                    else:
                        s += embedding[emb['UNK']]
                    word_count += 1
            if word_count > 0:
                if domain:
                    return np.append(s / float(word_count), domain)
                else:
                    return s / float(word_count)
            return None

        def read_test_data(directory):
            dataset = np.array([])
            stopwords = set([w.rstrip('\r\n') for w in open('stopwords.txt')])
            files = os.listdir(directory)
            for i in tqdm.tqdm(range(len(files))):
                if files[i].endswith(".txt"):
                    with open(os.path.join(directory, files[i]), encoding='utf8') as file:
                        data = []
                        for line in file.readlines():
                            split = preprocess_text(line, stopwords)  # split is a list of sen
                            for sentence in split:
                                if sentence:
                                    data += [sentence]
                        val = embeddings_mean(embedding, emb, data)
                        if val is not None:
                            if dataset.size == 0:
                                dataset = np.append(dataset, val)
                            else:
                                dataset = np.vstack((dataset, val))
            return dataset

In [4]: TRAIN_DIR = "dataset/DATA/TRAIN"
        VALID_DIR = "dataset/DATA/DEV"
        TEST_DIR = "dataset/DATA/TEST"
        TMP_DIR = "tmp/"
        if os.path.exists(TMP_DIR + 'train.pkl'):
            # if the train dataset has already been stored in train.pkl then load it
            train = pickle.load(open(TMP_DIR + 'train.pkl', 'rb'))
        else:
            # otherwise read the train dataset and store it in train.pkl
            train = read_data(TRAIN_DIR)
            pickle.dump(train, open(TMP_DIR + 'train.pkl', 'wb'))

        if os.path.exists(TMP_DIR + 'dev.pkl'):
            # if the validation dataset has already been stored in dev.pkl then load it
            validation = pickle.load(open(TMP_DIR + 'dev.pkl', 'rb'))
        else:
            # otherwise read the validation dataset and store it in dev.pkl
            validation = read_data(VALID_DIR)
            pickle.dump(validation, open(TMP_DIR + 'dev.pkl', 'wb'))
```

```
In [5]: train.shape
```

```
Out[5]: (114563, 129)
```

```
In [6]: validation.shape
```

```
Out[6]: (24546, 129)
```

```
In [7]: X_train = train[:, :-1].astype(np.float32)
        Y_train = train[:, train.shape[1] - 1]
        del train

        X_validation = validation[:, :-1].astype(np.float32)
        Y_validation = validation[:, validation.shape[1] - 1]
        del validation
```

```
In [8]: from sklearn.model_selection import KFold
        from sklearn.model_selection import cross_val_score
        from sklearn.metrics import classification_report
        from sklearn.metrics import confusion_matrix
        from sklearn.metrics import accuracy_score
        from sklearn.linear_model import LogisticRegression
        from sklearn.tree import DecisionTreeClassifier
        from sklearn.neighbors import KNeighborsClassifier
        from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
        from sklearn.naive_bayes import GaussianNB
```

```
In [9]: # Test to choose the best classification algorithm
        models = []
        models.append(('LR', LogisticRegression()))
        models.append(('LDA', LinearDiscriminantAnalysis()))
        models.append(('KNN', KNeighborsClassifier()))
        models.append(('CART', DecisionTreeClassifier()))
        models.append(('NB', GaussianNB()))

        # evaluation for each model
        for name, model in models:
            kfold = KFold()
            cv_results = cross_val_score(model, X_train, Y_train, cv=kfold, scoring='accuracy'
            msg = "%s: %f (%f)" % (name, cv_results.mean(), cv_results.std())
            print(msg)
```

```
LR: 0.144731 (0.124815)
LDA: 0.135462 (0.107099)
KNN: 0.140838 (0.117109)
CART: 0.101681 (0.081302)
NB: 0.080645 (0.059856)
```

```
In [10]: # KNN returns the highest accuracy score
         model = KNeighborsClassifier()
         model.fit(X_train, Y_train)
         predictions = model.predict(X_validation)
         array = confusion_matrix(Y_validation, predictions)
         print(classification_report(Y_validation, predictions))
         print('Accuracy: %.2f' % (accuracy_score(Y_validation, predictions) * 100) + '%')
```

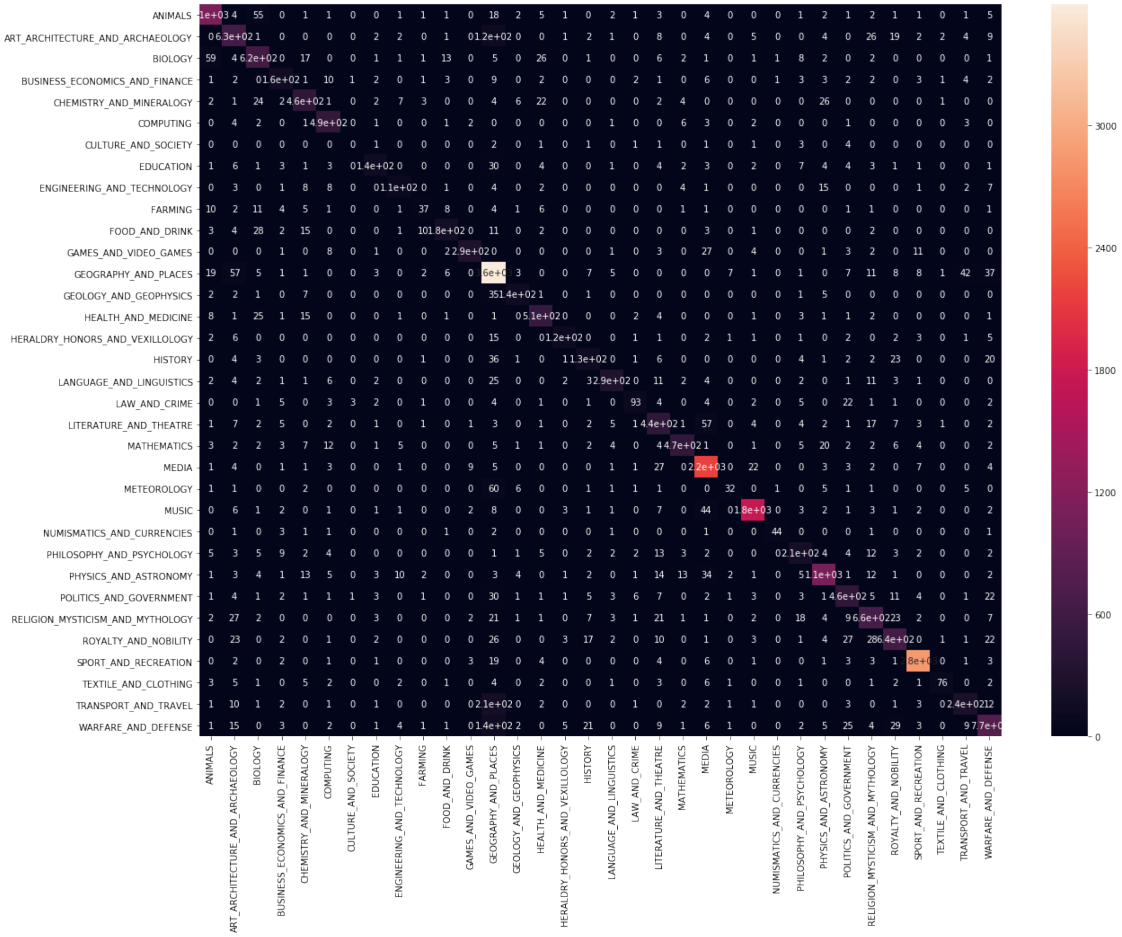
```
                                  precision    recall  f1-score   support

                         ANIMALS       0.90      0.91      0.90      1241
ART_ARCHITECTURE_AND_ARCHAEOLOGY       0.74      0.75      0.75       841
                         BIOLOGY       0.78      0.80      0.79       776
  BUSINESS_ECONOMICS_AND_FINANCE       0.73      0.71      0.72       217
        CHEMISTRY_AND_MINERALOGY       0.81      0.81      0.81       569
                       COMPUTING       0.86      0.95      0.90       515
             CULTURE_AND_SOCIETY       0.00      0.00      0.00        16
                       EDUCATION       0.80      0.63      0.71       222
      ENGINEERING_AND_TECHNOLOGY       0.75      0.66      0.70       167
                         FARMING       0.60      0.39      0.47        95
                  FOOD_AND_DRINK       0.81      0.68      0.74       258
           GAMES_AND_VIDEO_GAMES       0.94      0.82      0.87       354
            GEOGRAPHY_AND_PLACES       0.81      0.94      0.87      3827
          GEOLOGY_AND_GEOPHYSICS       0.83      0.71      0.77       192
             HEALTH_AND_MEDICINE       0.85      0.88      0.87       577
 HERALDRY_HONORS_AND_VEXILLOLOGY       0.87      0.74      0.80       166
                         HISTORY       0.63      0.55      0.58       232
        LANGUAGE_AND_LINGUISTICS       0.89      0.78      0.83       376
                   LAW_AND_CRIME       0.81      0.60      0.69       155
          LITERATURE_AND_THEATRE       0.72      0.77      0.74       571
                     MATHEMATICS       0.92      0.83      0.87       564
                           MEDIA       0.91      0.96      0.93      2273
                     METEOROLOGY       0.70      0.27      0.39       119
                           MUSIC       0.97      0.95      0.96      1861
      NUMISMATICS_AND_CURRENCIES       0.94      0.77      0.85        57
       PHILOSOPHY_AND_PSYCHOLOGY       0.72      0.71      0.71       294
           PHYSICS_AND_ASTRONOMY       0.90      0.89      0.90      1223
         POLITICS_AND_GOVERNMENT       0.78      0.79      0.78       580
RELIGION_MYSTICISM_AND_MYTHOLOGY       0.81      0.81      0.81       823
            ROYALTY_AND_NOBILITY       0.82      0.79      0.80       811
            SPORT_AND_RECREATION       0.98      0.98      0.98      2899
            TEXTILE_AND_CLOTHING       0.92      0.64      0.75       119
            TRANSPORT_AND_TRAVEL       0.77      0.49      0.60       495
             WARFARE_AND_DEFENSE       0.82      0.73      0.77      1061

                     avg / total       0.86      0.86      0.85     24546

Accuracy: 85.72%
```

```
In [11]: conf_mat = pd.DataFrame(array,
                              index = [domain for domain in os.listdir(TRAIN_DIR)],
                              columns = [domain for domain in os.listdir(TRAIN_DIR)])
         plt.figure(figsize = (20, 15))
         sn.heatmap(conf_mat, annot=True)
         plt.savefig('conf_mat.png')
```



	ANIMALS	ART_ARCHITECTURE_AND_ARCHAEOLOGY	BIOLOGY	BUSINESS_ECONOMICS_AND_FINANCE	CHEMISTRY_AND_MINERALOGY	COMPUTING	CULTURE_AND_SOCIETY	EDUCATION	ENGINEERING_AND_TECHNOLOGY	FARMING	FOOD_AND_DRINK	GAMES_AND_VIDEO_GAMES	GEOGRAPHY_AND_PLACES	GEOLOGY_AND_GEOPHYSICS	HEALTH_AND_MEDICINE	HERALDRY_HONORS_AND_VEXILLOLOGY	HISTORY	LANGUAGE_AND_LINGUISTICS	LAW_AND_CRIME	LITERATURE_AND_THEATRE	MATHEMATICS	MEDIA	METEOROLOGY	MUSIC	NUMISMATICS_AND_CURRENCIES	PHILOSOPHY_AND_PSYCHOLOGY	PHYSICS_AND_ASTRONOMY	POLITICS_AND_GOVERNMENT	RELIGION_MYSTICISM_AND_MYTHOLOGY	ROYALTY_AND_NOBILITY	SPORT_AND_RECREATION	TEXTILE_AND_CLOTHING	TRANSPORT_AND_TRAVEL	WARFARE_AND_DEFENSE
ANIMALS	1e+03	4	55	0	1	1	0	0	1	1	1	0	18	2	5	1	0	2	1	3	0	4	0	0	0	1	2	1	2	1	1	0	1	5
ART_ARCHITECTURE_AND_ARCHAEOLOGY	0	6.3e+02	1	0	0	0	0	2	2	0	1	0	1.2e+02	0	0	1	2	1	0	8	0	4	0	5	0	0	4	0	26	19	2	2	4	9
BIOLOGY	59	4	6.2e+02	0	17	0	0	1	1	1	13	0	5	0	26	0	1	0	0	6	2	1	0	1	1	8	2	0	2	0	0	0	0	1
BUSINESS_ECONOMICS_AND_FINANCE	1	2	0	1.6e+02	1	10	1	2	0	1	3	0	9	0	2	0	0	0	2	1	0	6	0	0	1	3	3	2	2	0	3	1	4	2
CHEMISTRY_AND_MINERALOGY	2	1	24	2	4.6e+02	1	0	2	7	3	0	0	4	6	22	0	0	0	0	2	4	0	0	0	0	0	26	0	0	0	0	1	0	0
COMPUTING	0	4	2	0	1	4.9e+02	0	1	0	0	1	2	0	0	0	0	0	1	0	0	6	3	0	2	0	0	0	1	0	0	0	0	3	0
CULTURE_AND_SOCIETY	0	0	0	0	0	0	0	1	0	0	0	0	2	0	1	0	1	0	1	1	0	1	0	1	0	3	0	4	0	0	0	0	0	0
EDUCATION	1	6	1	3	1	3	0	1.4e+02	0	0	0	0	30	0	4	0	0	1	0	4	2	3	0	2	0	7	4	4	3	1	1	0	0	1
ENGINEERING_AND_TECHNOLOGY	0	3	0	1	8	8	0	0	1.1e+02	0	1	0	4	0	2	0	0	0	0	0	4	1	0	0	0	0	15	0	0	0	1	0	2	7
FARMING	10	2	11	4	5	1	0	0	1	37	8	0	4	1	6	0	0	0	0	0	1	1	0	0	0	0	0	1	1	0	0	0	0	1
FOOD_AND_DRINK	3	4	28	2	15	0	0	0	1	10	1.8e+02	0	11	0	2	0	0	0	0	0	0	3	0	1	0	0	0	0	2	0	0	0	0	0
GAMES_AND_VIDEO_GAMES	0	0	0	1	0	8	0	1	0	0	2	2.9e+02	0	0	0	0	0	1	0	3	0	27	0	4	0	0	1	3	2	0	11	0	0	0
GEOGRAPHY_AND_PLACES	19	57	5	1	1	0	0	3	0	2	6	0	3.6e+03	3	0	0	7	5	0	0	0	0	7	1	0	1	0	7	11	8	8	1	42	37
GEOLOGY_AND_GEOPHYSICS	2	2	1	0	7	0	0	0	0	0	0	0	35	1.4e+02	1	0	1	0	0	0	0	0	0	0	0	1	5	0	0	0	0	0	0	0
HEALTH_AND_MEDICINE	8	1	25	1	15	0	0	0	1	0	1	0	1	0	5.1e+02	0	0	0	2	4	0	0	0	1	0	3	1	1	2	0	0	0	0	1
HERALDRY_HONORS_AND_VEXILLOLOGY	2	6	0	0	0	0	0	0	0	0	0	0	15	0	0	1.2e+02	0	0	1	1	0	2	1	1	0	1	0	2	0	2	3	0	1	5
HISTORY	0	4	3	0	0	0	0	0	0	1	0	0	36	1	0	1	1.3e+02	0	1	6	0	0	0	0	0	4	1	2	2	23	0	0	0	20
LANGUAGE_AND_LINGUISTICS	2	4	2	1	1	6	0	2	0	0	0	0	25	0	0	2	3	2.9e+02	0	11	2	4	0	0	0	2	0	1	11	3	1	0	0	0
LAW_AND_CRIME	0	0	1	5	0	3	3	2	0	1	0	0	4	0	1	0	1	0	93	4	0	4	0	2	0	5	0	22	1	1	0	0	0	2
LITERATURE_AND_THEATRE	1	7	2	5	0	2	0	1	0	1	0	1	3	0	1	0	2	5	1	4.4e+02	1	57	0	4	0	4	2	1	17	7	3	1	0	2
MATHEMATICS	3	2	2	3	7	12	0	1	5	0	0	0	5	1	1	0	2	4	0	4	4.7e+02	1	0	1	0	5	20	2	2	6	4	0	0	2
MEDIA	1	4	0	1	1	3	0	0	1	0	0	9	5	0	0	0	0	1	1	27	0	2.2e+03	0	22	0	0	3	3	2	0	7	0	0	4
METEOROLOGY	1	1	0	0	2	0	0	0	0	0	0	0	60	6	0	0	1	1	1	1	0	0	32	0	1	0	5	1	1	0	0	0	5	0
MUSIC	0	6	1	2	0	1	0	1	1	0	0	2	8	0	0	3	1	1	0	7	0	44	0	1.8e+03	0	3	2	1	3	1	2	0	0	2
NUMISMATICS_AND_CURRENCIES	0	1	0	3	1	1	0	0	0	0	1	0	2	0	0	0	0	1	0	0	0	1	0	0	44	0	0	0	1	0	0	0	0	1
PHILOSOPHY_AND_PSYCHOLOGY	5	3	5	9	2	4	0	0	0	0	0	0	1	1	5	0	2	2	2	13	3	2	0	0	0	2.1e+02	4	4	12	3	2	0	0	2
PHYSICS_AND_ASTRONOMY	1	3	4	1	13	5	0	3	10	2	0	0	3	4	0	1	2	0	1	14	13	34	2	1	0	5	1.1e+03	1	12	1	0	0	0	2
POLITICS_AND_GOVERNMENT	1	4	1	2	1	1	1	3	0	1	0	0	30	1	1	1	5	3	6	7	0	2	1	3	0	3	1	4.6e+02	5	11	4	0	1	22
RELIGION_MYSTICISM_AND_MYTHOLOGY	2	27	2	0	0	0	0	3	0	0	0	2	21	1	1	0	7	3	1	21	1	1	0	2	0	18	4	9	6.6e+02	23	2	0	0	7
ROYALTY_AND_NOBILITY	0	23	0	2	0	1	0	2	0	0	0	0	26	0	0	3	17	2	0	10	0	1	0	3	0	1	4	27	28	6.4e+02	0	1	1	22
SPORT_AND_RECREATION	0	2	0	2	0	1	0	1	0	0	0	3	19	0	4	0	0	0	0	4	0	6	0	1	0	0	1	3	3	1	2.8e+03	0	1	3
TEXTILE_AND_CLOTHING	3	5	1	0	5	2	0	0	2	0	1	0	4	0	2	0	0	1	0	3	0	6	1	0	0	1	0	0	1	2	1	76	0	2
TRANSPORT_AND_TRAVEL	1	10	1	2	0	1	0	1	0	0	0	0	2.1e+02	0	2	0	0	0	1	0	2	2	1	1	0	0	0	3	0	1	3	0	2.4e+02	12
WARFARE_AND_DEFENSE	1	15	0	3	0	2	0	1	4	1	1	0	1.4e+02	2	0	5	21	0	0	9	1	6	1	0	0	2	5	25	4	29	3	0	9	7.7e+0

```
In [12]: if os.path.exists(TMP_DIR + 'test.pkl'):
             # if the test dataset has already been stored in test.pkl then load it
             test = pickle.load(open(TMP_DIR + 'test.pkl', 'rb'))
         else:
             # otherwise read the test dataset and store it in test.pkl
             test = read_test_data(TEST_DIR)
             pickle.dump(train, open(TMP_DIR + 'test.pkl', 'wb'))

In [13]: test.shape

Out[13]: (24526, 128)

In [14]: test_predictions = model.predict(test)

In [15]: with open('test_answers.tsv', 'w') as f:
             for i in range(len(test_predictions)):
                 f.write('test_' + str(i) + '\t' + test_predictions[i] + '\n')
```