Classifiers1551

August 4, 2019

```
In [319]: import pandas as pd
          from tqdm import tqdm
          import os
          import sys
          import tokenize_uk
          from langdetect import detect
          import json
          import os.path
          import pickle
          import numpy as np
          from matplotlib import pylab
          from six.moves import range
          from sklearn.manifold import TSNE
          from sklearn import decomposition
          from sklearn.model_selection import train_test_split
          from collections import Counter

In [3]: INTERMEDIATE_RESULTS_PATH = "cleared_texts.csv"
        INTERMEDIATE_RESULTS = os.path.isfile(INTERMEDIATE_RESULTS_PATH)

In [4]: import gensim
        model = gensim.models.KeyedVectors.load_word2vec_format('ubercorpus.cased.tokenized.wo

In [5]: TEXT_DATA_DIR = '1551'

In [6]: stopwords = []

        with open('ukrainian-stopwords.txt','r') as f:
            for line in f:
                for word in line.split():
                    stopwords.append(word)

In [7]: def parse_raw_data():

            texts = []

            for fname in tqdm(sorted(os.listdir(TEXT_DATA_DIR))):
                fpath = os.path.join(TEXT_DATA_DIR, fname)
                f = open(fpath, encoding='utf-8')
                t = f.read()
                res = [x.split("\n", 1) for x in t.split("\n\n\n")]
                for z in res:
                    if (len(z) == 2):
                        try:
                            if (detect(z[1]) != 'uk'):
                                # Filter out non-ukrainian texts.
                                continue
                            text = tokenize_uk.tokenize_text(z[1])
                            text = [item for sublist in text for item in sublist]
                            text = [item for sublist in text for item in sublist]
                            text = list(filter(lambda x: len(x) > 2 and x not in stopwords and
                            # flattening needs to be done twice because text may have multiple
                            texts.append({"key": z[0], "text": text, "label": fname})
                        except:
                            continue
                f.close()

            print('Found %s texts.' % len(texts))

            res = pd.DataFrame(texts)
            res.to_csv(INTERMEDIATE_RESULTS_PATH)
```

```
In [404]: if not INTERMEDIATE_RESULTS:
              parse_raw_data()

          texts = pd.read_csv("cleared_texts.csv")
          texts = texts[texts["text"] != "[]"]
```

```
In [405]: summarized_vectors = {}
          failed_keys = []
          for pos, row in tqdm(texts.iterrows()):
              try:
                  words = []
                  strfromcsv = "words = " + row["text"]
                  exec(strfromcsv)
                  summarized_vectors[row["key"]] = np.sum(model[words], axis=0)
              except:
                  failed_keys.append(row["key"])

          texts = texts[~texts.text.isin(failed_keys)]
```

```
62425it [01:00, 1027.81it/s]
```

```
In [417]: from numpy import dot
          from numpy.linalg import norm
          from sklearn import cluster
```

```
         from sklearn import metrics
         from sklearn.neighbors import KNeighborsClassifier

In [476]: class KNN_Classifer():

              def __init__(self, x_train):
                  self.x_train = x_train

              def fit(self, verbose = 0):
                  x_train = self.x_train
                  X = [summarized_vectors.get(key) for key in x_train['key']]
                  y = x_train['label']
                  classifier = KNeighborsClassifier(n_neighbors=5, metric='cosine')
                  classifier.fit(X, y)
                  return classifier

In [477]: X_train, X_test = train_test_split(texts, test_size=0.3)

In [478]: knn = KNN_Classifer(X_train)
          classifier = knn.fit()
```

```
Class instance initialized.
```

```
In [484]: def predict(x_test, c):
                  y_predict = []

                  for i in tqdm(range(x_test.shape[0])):
                      lbl = c.predict([summarized_vectors[X_test.iloc[i]['key']]])[0]
                      y_predict.append(lbl)

                  return y_predict

In [485]: y = predict(X_test, classifier)
```

```
  0%|          | 0/18728 [00:00<?, ?it/s]
  0%|          | 1/18728 [00:00<4:59:16,  1.04it/s]
  0%|          | 2/18728 [00:01<4:14:08,  1.23it/s]
  0%|          | 3/18728 [00:01<3:20:37,  1.56it/s]
  0%|          | 4/18728 [00:01<2:39:00,  1.96it/s]
  0%|          | 5/18728 [00:02<2:09:32,  2.41it/s]
  0%|          | 6/18728 [00:02<1:48:44,  2.87it/s]
  0%|          | 7/18728 [00:02<1:36:58,  3.22it/s]
  0%|          | 8/18728 [00:02<1:23:28,  3.74it/s]
  0%|          | 9/18728 [00:02<1:12:58,  4.27it/s]
  0%|          | 10/18728 [00:02<1:07:40,  4.61it/s]
  0%|          | 11/18728 [00:03<1:01:19,  5.09it/s]
  0%|          | 12/18728 [00:03<56:37,  5.51it/s]
  0%|          | 13/18728 [00:03<54:32,  5.72it/s]
  0%|          | 14/18728 [00:03<54:19,  5.74it/s]
  0%|          | 15/18728 [00:03<52:50,  5.90it/s]
  0%|          | 16/18728 [00:03<54:15,  5.75it/s]
  0%|          | 17/18728 [00:04<59:12,  5.27it/s]
  0%|          | 18/18728 [00:04<1:06:19,  4.70it/s]
  0%|          | 19/18728 [00:04<1:01:03,  5.11it/s]
  0%|          | 20/18728 [00:04<59:29,  5.24it/s]
  0%|          | 21/18728 [00:05<1:01:56,  5.03it/s]
  0%|          | 22/18728 [00:05<1:04:23,  4.84it/s]
  0%|          | 23/18728 [00:05<1:06:55,  4.66it/s]
  0%|          | 24/18728 [00:05<1:01:05,  5.10it/s]
  0%|          | 25/18728 [00:05<58:26,  5.33it/s]
  0%|          | 26/18728 [00:05<58:07,  5.36it/s]
  0%|          | 27/18728 [00:06<56:04,  5.56it/s]
  0%|          | 28/18728 [00:06<56:17,  5.54it/s]
  0%|          | 29/18728 [00:06<54:31,  5.72it/s]
  0%|          | 30/18728 [00:06<52:15,  5.96it/s]
  0%|          | 31/18728 [00:06<51:36,  6.04it/s]
  0%|          | 32/18728 [00:06<51:51,  6.01it/s]
Exception in thread Thread-85:
Traceback (most recent call last):
  File "/Library/Frameworks/Python.framework/Versions/3.6/lib/python3.6/threading.py", line 91
    self.run()
  File "/Library/Frameworks/Python.framework/Versions/3.6/lib/python3.6/site-packages/tqdm/_tq
    for instance in self.tqdm_cls._instances:
  File "/Library/Frameworks/Python.framework/Versions/3.6/lib/python3.6/_weakrefset.py", line 
    for itemref in self.data:
RuntimeError: Set changed size during iteration

100%|| 18728/18728 [31:11<00:00, 10.01it/s]
```

```
In [492]: def calculate_error_rate(predicted):
              error_rate = sum(1 for i, j in zip(predicted, X_test['label']) if i != j) / len(
              return error_rate
```

```
In [487]: class SVM_Classifer():

              def __init__(self, x_train):
                  self.x_train = x_train

              def fit(self, verbose = 0):
                  x_train = self.x_train
                  X = [summarized_vectors.get(key) for key in x_train['key']]
                  y = x_train['label']
                  clf = svm.SVC(gamma='scale')
                  classifier = svm.SVC(gamma='scale')
                  classifier.fit(X, y)
                  return classifier
```

```
In [490]: svm = SVM_Classifer(X_train)
          classifier_svm = svm.fit()
```

```
In [491]: y_svm = predict(X_test, classifier_svm)
```

```
100%|| 18728/18728 [26:17<00:00, 11.88it/s]
```

```
In [497]: print("SVM error rate: " + str(calculate_error_rate(y_svm)))
```

```
SVM error rate: 0.5866616830414353
```

```
In [496]: print("KNN error rate: " + str(calculate_error_rate(y)))
```

```
KNN error rate: 0.5866616830414353
```