Лабораторная работа №6 по курсу "Методы машинного обучения"

Выполнил: Саврасов П.А. группа ИУ5-24М

Задание

Для произвольного набора данных, предназначенного для классификации текстов, решите задачу классификации текста двумя способами:

1. На основе CountVectorizer или TfidfVectorizer.
2. На основе моделей word2vec или Glove или fastText.

Сравните качество полученных моделей.

In [48]:
```
import re
import pandas as pd
import numpy as np
from typing import Dict, Tuple
from sklearn.model_selection import train_test_split
from sklearn.pipeline import Pipeline
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score
from nltk import WordPunctTokenizer
from nltk.corpus import stopwords
import nltk
import gensim
from gensim.models import word2vec
nltk.download('stopwords')
```

```
/home/pavel/venvs/tensorflow/lib/python3.9/site-packages/gensim/similarities/__init__.py:15: UserWarning: The gensim.similarities.levenshtein submodule is disabled, because the optional Levenshtein package <https://pypi.org/project/python-Levenshtein/> is unavailable. Install Levenhstein (e.g. `pip install python-Levenshtein`) to suppress this warning.
  warnings.warn(msg)
[nltk_data] Downloading package stopwords to /home/pavel/nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
```

Out[48]:
```
True
```

In [49]:
```
data = pd.read_csv('Youtube04.csv', sep = ',')
data.head()
```

Out[49]:

	COMMENT_ID	AUTHOR	DATE	CONTENT	CLASS
0	z12rwfnyyrbsefonb232i5ehdxzkjzjs2	Lisa Wellas	NaN	+447935454150 lovely girl talk to me xxx	1
1	z130wpnwwnyuetxcn23xf5k5ynmkdpjrj04	jason graham	2015-05-29T02:26:10.652000	I always end up coming back to this song 	0
2	z13vsfqirtavjvu0t22ezrgzyorwxhpf3	Ajkal Khan	NaN	my sister just received over 6,500 new <a rel=...	1
3	z12wjzc4eprnvja4304cgbbizuved35wxcs	Dakota Taylor	2015-05-29T02:13:07.810000	Cool	0
4	z13xjfr42z3uxdz2223gx5rrzs3dt5hna	Jihad Naser	NaN	Hello I'am from Palastine	1

На основе CountVectorizer

In [50]:
```
content = data['CONTENT']
target = data['CLASS']

TrainX, TestX, TrainY, TestY = train_test_split(content, target, test_size=0.3, random_state = 1)
```

In [61]:
```
model = Pipeline(
        [("vectorizer", CountVectorizer()),
         ("classifier", RandomForestClassifier())])
model.fit(TrainX, TrainY)
print_accuracy_score_for_classes(TestY, model.predict(TestX))
```

```
Метка    Accuracy
0        0.9848484848484849
1        0.9130434782608695
```

На основе Word2vec

In [52]:
```
corpus = []
stop_words = stopwords.words('english')
tok = WordPunctTokenizer()
for line in data['CONTENT'].values:
    line1 = line.strip().lower()
    line1 = re.sub("[^a-zA-Z]"," ", line1)
    text_tok = tok.tokenize(line1)
    text_tok1 = [w for w in text_tok if not w in stop_words]
    corpus.append(text_tok1)
```

In [53]:
```
corpus[5:10]
```

Out[53]:
```
[['wow', 'video', 'almost', 'billion', 'views', 'know', 'popular'],
 ['go', 'check', 'rapping', 'video', 'called', 'four', 'wheels', 'please'],
 ['almost', 'billion'],
 ['aslamu', 'lykum', 'pakistan'],
 ['eminem', 'idol', 'people', 'espa', 'mexico', 'latinoamerica']]
```

In [54]:
```
model_imdb = word2vec.Word2Vec(corpus, workers=4, min_count=10, window=10, sample=1e-3)
```

In [55]:
```
class EmbeddingVectorizer(object):
    '''
    Для текста усредним вектора входящих в него слов
    '''
    def __init__(self, model):
        self.model = model
        self.size = model.vector_size

    def fit(self, X, y):
        return self

    def transform(self, X):
        return np.array([np.mean(
            [self.model[w] for w in words if w in self.model]
            or [np.zeros(self.size)], axis=0)
            for words in X])
```

In [56]:
```
def accuracy_score_for_classes(
    y_true: np.ndarray,
    y_pred: np.ndarray) -> Dict[int, float]:
    """
    Вычисление метрики accuracy для каждого класса
    y_true - истинные значения классов
    y_pred - предсказанные значения классов
    Возвращает словарь: ключ - метка класса,
    значение - Accuracy для данного класса
    """
    # Для удобства фильтрации сформируем Pandas DataFrame
    d = {'t': y_true, 'p': y_pred}
    df = pd.DataFrame(data=d)
    # Метки классов
    classes = np.unique(y_true)
    # Результирующий словарь
    res = dict()
    # Перебор меток классов
    for c in classes:
        # отфильтруем данные, которые соответствуют
        # текущей метке класса в истинных значениях
        temp_data_flt = df[df['t']==c]
        # расчет accuracy для заданной метки класса
        temp_acc = accuracy_score(
            temp_data_flt['t'].values,
            temp_data_flt['p'].values)
        # сохранение результата в словарь
        res[c] = temp_acc
    return res

def print_accuracy_score_for_classes(
    y_true: np.ndarray,
    y_pred: np.ndarray):
    """
    Вывод метрики accuracy для каждого класса
    """
    accs = accuracy_score_for_classes(y_true, y_pred)
    if len(accs)>0:
        print('Метка \t Accuracy')
    for i in accs:
        print('{} \t {}'.format(i, accs[i]))
```

In [60]:
```
model2 = Pipeline(
        [("vectorizer", EmbeddingVectorizer(model_imdb.wv)),
         ("classifier", RandomForestClassifier())])
model2.fit(TrainX,TrainY)
print_accuracy_score_for_classes(TestY, model2.predict(TestX))
```

```
Метка    Accuracy
0        0.7575757575757576
1        0.7391304347826086
```

In [ ]:
```

```