Московский государственный технический университет им. Н.Э. Баумана
Факультет «Информатика и системы управления»
Кафедра «Системы обработки информации и управления»

Лабораторная работа №6
по дисциплине «Методы машинного обучения»
«Классификация текста»

ИСПОЛНИТЕЛЬ:

Цветкова Алена
Группа ИУ5-21М

____

ПРЕПОДАВАТЕЛЬ:

Гапанюк Ю.Е.

____

Москва, 2021

Цель лабораторной работы: изучение методов классификации текстов.

In [1]:

```
import pandas as pd
import numpy as np
from typing import Dict, Tuple
from sklearn.pipeline import Pipeline
from sklearn.feature_extraction.text import CountVectorizer, TfidfVectoriz
er
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, balanced_accuracy_score
from sklearn.svm import SVC, NuSVC, LinearSVC, OneClassSVM, SVR, NuSVR, Li
nearSVR
from sklearn.neighbors import KNeighborsRegressor, KNeighborsClassifier
from nltk import WordPunctTokenizer
from nltk.corpus import stopwords
import nltk
import re
from gensim.models import word2vec
nltk.download('stopwords')
```

```
//anaconda3/lib/python3.7/site-packages/gensim/similarities/_
_init__.py:15: UserWarning: The gensim.similarities.levenshte
in submodule is disabled, because the optional Levenshtein pa
ckage <https://pypi.org/project/python-Levenshtein/> is unava
ilable. Install Levenhstein (e.g. `pip install python-Levensh
tein`) to suppress this warning.
  warnings.warn(msg)
[nltk_data] Downloading package stopwords to
[nltk_data]     /Users/alena.tsvetkova/nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
```

Out[1]:

```
True
```

In [2]:

```
test_data = pd.read_csv('archive/SMS_test.csv', encoding= 'unicode_escape'
)
train_data = pd.read_csv('archive/SMS_train.csv', encoding= 'unicode_escap
e')
```

In [3]:

```
test_data.head()
```

Out[3]:

	S. No.	Message_body	Label
0	1	UpgrdCentre Orange customer, you may now claim...	Spam
1	2	Loan for any purpose £500 - £75,000. Homeowner...	Spam
2	3	Congrats! Nokia 3650 video camera phone is you...	Spam
3	4	URGENT! Your Mobile number has been awarded wi...	Spam

3	4	URGENT! Your Mobile number has been awarded wi...	Spam
4	5	Someone has contacted our dating service and e...	Spam

In [4]:

```
train_data.head()
```

Out[4]:

	S. No.	Message_body	Label
0	1	Rofl. Its true to its name	Non-Spam
1	2	The guy did some bitching but I acted like i'd...	Non-Spam
2	3	Pity, * was in mood for that. So...any other s...	Non-Spam
3	4	Will ü b going to esplanade fr home?	Non-Spam
4	5	This is the 2nd time we have tried 2 contact u...	Spam

In [23]:

```
X_train = train_data['Message_body']
X_test = test_data['Message_body']
y_train = train_data['Label']
y_test = test_data['Label']
```

In [6]:

```
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
```

In [7]:

```
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]))
```

Способ 1. На основе CountVectorizer или TfidfVectorizer.

In [8]:

```
def sentiment(v, c):
    model = Pipeline(
        [("vectorizer", v),
         ("classifier", c)])
    model.fit(X_train, y_train)
    y_pred = model.predict(X_test)
    print_accuracy_score_for_classes(y_test, y_pred)
```

In [9]:

```
sentiment(CountVectorizer(), LogisticRegression(C=5.0))
```

```
Метка    Accuracy
Non-Spam         1.0
Spam     0.75
```

```
//anaconda3/lib/python3.7/site-packages/sklearn/linear_model/
logistic.py:432: FutureWarning: Default solver will be change
d to 'lbfgs' in 0.22. Specify a solver to silence this warnin
g.
  FutureWarning)
```

In [10]:

```
sentiment(CountVectorizer(), LinearSVC())
```

```
Метка    Accuracy
Non-Spam         1.0
Spam     0.8026315789473685
```

In [11]:

```
sentiment(CountVectorizer(), KNeighborsClassifier())
```

```
Метка    Accuracy
Non-Spam         1.0
Spam     0.10526315789473684
```

In [12]:

```
sentiment(TfidfVectorizer(), LogisticRegression(C=5.0))
```

```
Метка     Accuracy

Non-Spam         1.0
Spam     0.6710526315789473
```

```
//anaconda3/lib/python3.7/site-packages/sklearn/linear_model/
logistic.py:432: FutureWarning: Default solver will be change
d to 'lbfgs' in 0.22. Specify a solver to silence this warnin
g.
  FutureWarning)
```

In [13]:

```
sentiment(TfidfVectorizer(), LinearSVC())
```

```
Метка     Accuracy
Non-Spam         1.0
Spam     0.7894736842105263
```

In [14]:

```
sentiment(TfidfVectorizer(), KNeighborsClassifier())
```

```
Метка     Accuracy
Non-Spam         0.9795918367346939
Spam     0.5394736842105263
```

Способ 2. На основе моделей word2vec или Glove или fastText.

In [15]:

```
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 [16]:

```
corpus_train = []
stop_words = stopwords.words('english')
tok = WordPunctTokenizer()
for line in train_data['Message_body'].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_train.append(text_tok1)
```

In [17]:

```
corpus_test = []
stop_words = stopwords.words('english')
tok = WordPunctTokenizer()
for line in test_data['Message_body'].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_test.append(text_tok1)
```

In [18]:

```
model_imdb = word2vec.Word2Vec(corpus_train, workers=4, min_count=10, wind
ow=10, sample=1e-3)
```

In [19]:

```
X_train = corpus_train
X_test = corpus_test
```

In [20]:

```
sentiment(EmbeddingVectorizer(model_imdb.wv), LogisticRegression(C=5.0))
```

```
Метка     Accuracy
Non-Spam          1.0
Spam      0.0
```

```
//anaconda3/lib/python3.7/site-packages/sklearn/linear_model/
logistic.py:432: FutureWarning: Default solver will be change
d to 'lbfgs' in 0.22. Specify a solver to silence this warnin
g.
  FutureWarning)
```

In [21]:

```
sentiment(EmbeddingVectorizer(model_imdb.wv), KNeighborsClassifier())
```

```
Метка     Accuracy
Non-Spam          1.0
Spam      0.7763157894736842
```

In [22]:

```
sentiment(EmbeddingVectorizer(model_imdb.wv), LinearSVC())
```

```
Метка     Accuracy
Non-Spam          1.0
Spam      0.0
```

Модель с максимальной точностью

In [24]:

```
sentiment(CountVectorizer(), LinearSVC())
```

```
Метка    Accuracy
Non-Spam         1.0
Spam     0.8026315789473685
```