Лабораторная работа №6

Писарчук Надежда ИУ5-22М

Тема: Классификация текста.

Задание:

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

Способ 1. На основе CountVectorizer или TfidfVectorizer.

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

Сравните качество полученных моделей. Для поиска наборов данных в поисковой системе можно использовать ключевые слова "datasets for text classification".

In [2]:

```
# This Python 3 environment comes with many helpful analytics libraries insta
# It is defined by the kaggle/python Docker image: https://github.com/kaggle/
# For example, here's several helpful packages to load

import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)

from typing import Dict, Tuple
from scipy import stats
from sklearn.preprocessing import LabelEncoder
from sklearn.feature_extraction.text import CountVectorizer, TfidfVectorizer
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsRegressor, KNeighborsClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import GridSearchCV, RandomizedSearchCV
from sklearn.metrics import accuracy_score, balanced_accuracy_score
from sklearn.metrics import precision_score, recall_score, f1_score, classifi
from sklearn.metrics import confusion_matrix
from sklearn.model_selection import cross_val_score
from sklearn.pipeline import Pipeline
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import mean_absolute_error, mean_squared_error, mean_squ
from sklearn.metrics import roc_curve, roc_auc_score
from sklearn.svm import SVC, NuSVC, LinearSVC, OneClassSVM, SVR, NuSVR, Linea
from sklearn.naive_bayes import ComplementNB

import nltk
from nltk.stem import WordNetLemmatizer
from nltk.corpus import wordnet
from nltk.corpus import stopwords

import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
sns.set(style="ticks")

# Input data files are available in the read-only "../input/" directory
# For example, running this (by clicking run or pressing Shift+Enter) will li

import os
for dirname, _, filenames in os.walk('/kaggle/input'):
    for filename in filenames:
        print(os.path.join(dirname, filename))

# You can write up to 20GB to the current directory (/kaggle/working/) that g
# You can also write temporary files to /kaggle/temp/, but they won't be save

pd.set_option("display.max_columns", None)
```

```
/kaggle/input/covid-19-nlp-text-classification/Corona_NLP_test.csv
/kaggle/input/covid-19-nlp-text-classification/Corona_NLP_train.csv
```

In [3]:
```
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 [4]:
```
train = pd.read_csv('/kaggle/input/covid-19-nlp-text-classification/Corona_NL
test = pd.read_csv('/kaggle/input/covid-19-nlp-text-classification/Corona_NLP
```

In [5]:
```
print(train.shape)
print(test.shape)
```

```
(41157, 6)
(3798, 6)
```

In [6]:
```
train.head()
```

Out[6]:

	UserName	ScreenName	Location	TweetAt	OriginalTweet	Sentiment
0	3799	48751	London	16-03-2020	@MeNyrbie @Phil_Gahan @Chrisitv https://t.co/i...	Neutral
1	3800	48752	UK	16-03-2020	advice Talk to your neighbours family to excha...	Positive
2	3801	48753	Vagabonds	16-03-2020	Coronavirus Australia: Woolworths to give elde...	Positive
3	3802	48754	NaN	16-03-2020	My food stock is not the only one which is emp...	Positive
4	3803	48755	NaN	16-03-2020	Me, ready to go at supermarket during the #COV...	Extremely Negative

In [7]:
```
train.Sentiment.value_counts()
```

Out[7]:
```
Positive              11422
Negative               9917
Neutral                7713
Extremely Positive     6624
Extremely Negative     5481
Name: Sentiment, dtype: int64
```

In [8]:
```
train.Sentiment = train.Sentiment.replace({'Extremely Positive':'Positive','E
test.Sentiment = test.Sentiment.replace({'Extremely Positive':'Positive','Ext

lenc = LabelEncoder()
test.Sentiment = lenc.fit_transform(test.Sentiment)
train.Sentiment = lenc.fit_transform(train.Sentiment)
```

In [9]:
```
train.head()
```

Out[9]:

	UserName	ScreenName	Location	TweetAt	OriginalTweet	Sentiment
0	3799	48751	London	16-03-2020	@MeNyrbie @Phil_Gahan @Chrisitv https://t.co/i...	1
1	3800	48752	UK	16-03-2020	advice Talk to your neighbours family to excha...	2
2	3801	48753	Vagabonds	16-03-2020	Coronavirus Australia: Woolworths to give elde...	2
3	3802	48754	NaN	16-03-2020	My food stock is not the only one which is emp...	2
4	3803	48755	NaN	16-03-2020	Me, ready to go at supermarket during the #COV...	0

In [49]:
```
x_train = train['OriginalTweet']
y_train = train['Sentiment']
x_test = test['OriginalTweet']
y_test = test['Sentiment']
```

Очистка данных

In [50]:
```
import re
def preprocess_sentence(w):
    # отделение слов и знаков пунктуации пробелом
    # eg: "he is a boy." => "he is a boy ."
    w = w.lower()
    w = re.sub('\t\n', '', w)
    w = re.sub(r'http\S+', '', w)
    w = re.sub(r"([?.!,])", r" \1 ", w)
    w = re.sub(r'[" "]+', " ", w)

    # удаляем все кроме (a-z, A-Z, ".", "?", "!", ",")
    w = re.sub(r"[^a-zA-Za-яA-Я?.!,`']+", " ", w)

    w = w.strip()
    tokens = w.split(' ')

    stop_words = set(stopwords.words('english')) # remove stopwords
    tokens = [word for word in tokens if not word in stop_words]
    tokens = ' '.join(tokens)
    return tokens
```

In [51]:
```
x_train = x_train.apply(preprocess_sentence)
x_test = x_test.apply(preprocess_sentence)
```

In [52]:
```
# Сформируем общий словарь для обучения моделей из обучающей и тестовой выбор
vocab_list = x_train.tolist() + x_test.tolist()
print(len(vocab_list))
vocab_list[1:10]
```

```
44955
```

Out[52]:
```
['advice talk neighbours family exchange phone numbers create contact list ph
one numbers neighbours schools employer chemist gp set online shopping accoun
ts poss adequate supplies regular meds order',
 'coronavirus australia woolworths give elderly , disabled dedicated shopping
hours amid covid outbreak',
 'food stock one empty . . . please , panic , enough food everyone take need
. stay calm , stay safe . covid france covid covid coronavirus confinement co
nfinementotal confinementgeneral',
 ", ready go supermarket covid outbreak . i'm paranoid , food stock litteraly
empty . coronavirus serious thing , please , panic . causes shortage . . . co
ronavirusfrance restezchezvous stayathome confinement",
 'news region first confirmed covid case came sullivan county last week , peo
ple flocked area stores purchase cleaning supplies , hand sanitizer , food ,
toilet paper goods , tim dodson reports',
 "cashier grocery store sharing insights covid prove credibility commented i'
```

```
m civics class know i'm talking .",
 'supermarket today . buy toilet paper . rebel toiletpapercrisis covid',
 'due covid retail store classroom atlanta open walk business classes next tw
o weeks , beginning monday , march . continue process online phone orders nor
mal ! thank understanding !',
 'corona prevention , stop buy things cash use online payment methods corona
spread notes . also prefer online shopping home . time fight covid ? . govind
```

In [14]:
```
train['OriginalTweet'][0:10][4]
```

Out[14]:
```
"Me, ready to go at supermarket during the #COVID19 outbreak.\r\r\n\r\r\nNot
because I'm paranoid, but because my food stock is litteraly empty. The #coro
navirus is a serious thing, but please, don't panic. It causes shortage...\r\
r\n\r\r\n#CoronavirusFrance #restezchezvous #StayAtHome #confinement https://
t.co/usmuaLq72n"
```

In [15]:
```
vocabVect = CountVectorizer()
vocabVect.fit(vocab_list)
corpusVocab = vocabVect.vocabulary_
print('Количество сформированных признаков - {}'.format(len(corpusVocab)))
```

```
Количество сформированных признаков - 54546
```

In [16]:
```
for i in list(corpusVocab)[1:10]:
    print('{}={}'.format(i, corpusVocab[i]))
```

```
phil=36274
gahan=18888
chrisitv=8219
advice=670
talk=47397
neighbours=32568
family=16737
exchange=16272
phone=36318
```

In [17]:
```
tfidfv = TfidfVectorizer(ngram_range=(1,3))
tfidf_ngram_features = tfidfv.fit_transform(vocab_list)
tfidf_ngram_features
```

Out[17]:
```
<44955x1141506 sparse matrix of type '<class 'numpy.float64'>'
        with 2263622 stored elements in Compressed Sparse Row format>
```

In [18]:
```
def VectorizeAndClassify(vectorizers_list, classifiers_list):
    for v in vectorizers_list:
        for c in classifiers_list:
            pipeline1 = Pipeline([("vectorizer", v), ("classifier", c)])
            score = cross_val_score(pipeline1, x_train[:10000], y_train[:1000
            print('Векторизация - {}'.format(v))
            print('Модель для классификации - {}'.format(c))
            print('Accuracy = {}'.format(score))
            print('============================')
```

In [19]:

```
vectorizers_list = [CountVectorizer(vocabulary = corpusVocab), TfidfVectorize
classifiers_list = [RandomForestClassifier(), ComplementNB(), LogisticRegress
VectorizeAndClassify(vectorizers_list, classifiers_list)
```

```
Векторизация - CountVectorizer(vocabulary={'aa': 0, 'aaa': 1, 'aaaaakubosan':
2, 'aaaaas': 3,
                               'aaaand': 4, 'aaachatterjee': 5, 'aaanews': 6,
                               'aaannnddd': 7, 'aaanortheast': 8, 'aabutan': 9,
                               'aacopd': 10, 'aacounty': 11, 'aacountygovt': 12,
                               'aadeshrawal': 13, 'aadya': 14, 'aadyasitara': 1
5,
                               'aafp': 16, 'aahealth': 17, 'aahh': 18, 'aai': 1
9,
                               'aaisp': 20, 'aajeevika': 21, 'aajtak': 22,
                               'aakash': 23, 'aalonzowatt': 24, 'aalto': 25,
                               'aaltouniversity': 26, 'aalwajih': 27,
                               'aamaadmi': 28, 'aamaadmiparty': 29, ...})
Модель для классификации - RandomForestClassifier()
Accuracy = 0.6917000938246195
============================
Векторизация - CountVectorizer(vocabulary={'aa': 0, 'aaa': 1, 'aaaaakubosan':
2, 'aaaaas': 3,
                               'aaaand': 4, 'aaachatterjee': 5, 'aaanews': 6,
                               'aaannnddd': 7, 'aaanortheast': 8, 'aabutan': 9,
                               'aacopd': 10, 'aacounty': 11, 'aacountygovt': 12,
                               'aadeshrawal': 13, 'aadya': 14, 'aadyasitara': 1
5,
                               'aafp': 16, 'aahealth': 17, 'aahh': 18, 'aai': 1
9,
                               'aaisp': 20, 'aajeevika': 21, 'aajtak': 22,
                               'aakash': 23, 'aalonzowatt': 24, 'aalto': 25,
                               'aaltouniversity': 26, 'aalwajih': 27,
                               'aamaadmi': 28, 'aamaadmiparty': 29, ...})
Модель для классификации - ComplementNB()
Accuracy = 0.6462999629297063
============================
Векторизация - CountVectorizer(vocabulary={'aa': 0, 'aaa': 1, 'aaaaakubosan':
2, 'aaaaas': 3,
                               'aaaand': 4, 'aaachatterjee': 5, 'aaanews': 6,
                               'aaannnddd': 7, 'aaanortheast': 8, 'aabutan': 9,
                               'aacopd': 10, 'aacounty': 11, 'aacountygovt': 12,
                               'aadeshrawal': 13, 'aadya': 14, 'aadyasitara': 1
5,
                               'aafp': 16, 'aahealth': 17, 'aahh': 18, 'aai': 1
9,
                               'aaisp': 20, 'aajeevika': 21, 'aajtak': 22,
                               'aakash': 23, 'aalonzowatt': 24, 'aalto': 25,
                               'aaltouniversity': 26, 'aalwajih': 27,
                               'aamaadmi': 28, 'aamaadmiparty': 29, ...})
Модель для классификации - LogisticRegression(C=3.0, max_iter=1000)
Accuracy = 0.7155993643755497
============================
Векторизация - CountVectorizer(vocabulary={'aa': 0, 'aaa': 1, 'aaaaakubosan':
2, 'aaaaas': 3,
                               'aaaand': 4, 'aaachatterjee': 5, 'aaanews': 6,
                               'aaannnddd': 7, 'aaanortheast': 8, 'aabutan': 9,
                               'aacopd': 10, 'aacounty': 11, 'aacountygovt': 12,
                               'aadeshrawal': 13, 'aadya': 14, 'aadyasitara': 1
```

```
5,
                            'aafp': 16, 'aahealth': 17, 'aahh': 18, 'aai': 1
9,
                            'aaisp': 20, 'aajeevika': 21, 'aajtak': 22,
                            'aakash': 23, 'aalonzowatt': 24, 'aalto': 25,
                            'aaltouniversity': 26, 'aalwajih': 27,
                            'aamaadmi': 28, 'aamaadmiparty': 29, ...})
Модель для классификации - LinearSVC()
Accuracy = 0.7094994142485634
============================
Векторизация - TfidfVectorizer(vocabulary={'aa': 0, 'aaa': 1, 'aaaaakubosan':
2, 'aaaaas': 3,
                            'aaaand': 4, 'aaachatterjee': 5, 'aaanews': 6,
                            'aaannnddd': 7, 'aaanortheast': 8, 'aabutan': 9,
                            'aacopd': 10, 'aacounty': 11, 'aacountygovt': 12,
                            'aadeshrawal': 13, 'aadya': 14, 'aadyasitara': 1
5,
                            'aafp': 16, 'aahealth': 17, 'aahh': 18, 'aai': 1
9,
                            'aaisp': 20, 'aajeevika': 21, 'aajtak': 22,
                            'aakash': 23, 'aalonzowatt': 24, 'aalto': 25,
                            'aaltouniversity': 26, 'aalwajih': 27,
                            'aamaadmi': 28, 'aamaadmiparty': 29, ...})
Модель для классификации - RandomForestClassifier()
Accuracy = 0.6763995835696347
============================
Векторизация - TfidfVectorizer(vocabulary={'aa': 0, 'aaa': 1, 'aaaaakubosan':
2, 'aaaaas': 3,
                            'aaaand': 4, 'aaachatterjee': 5, 'aaanews': 6,
                            'aaannnddd': 7, 'aaanortheast': 8, 'aabutan': 9,
                            'aacopd': 10, 'aacounty': 11, 'aacountygovt': 12,
                            'aadeshrawal': 13, 'aadya': 14, 'aadyasitara': 1
5,
                            'aafp': 16, 'aahealth': 17, 'aahh': 18, 'aai': 1
9,
                            'aaisp': 20, 'aajeevika': 21, 'aajtak': 22,
                            'aakash': 23, 'aalonzowatt': 24, 'aalto': 25,
                            'aaltouniversity': 26, 'aalwajih': 27,
                            'aamaadmi': 28, 'aamaadmiparty': 29, ...})
Модель для классификации - ComplementNB()
Accuracy = 0.6464996829616975
============================
Векторизация - TfidfVectorizer(vocabulary={'aa': 0, 'aaa': 1, 'aaaaakubosan':
2, 'aaaaas': 3,
                            'aaaand': 4, 'aaachatterjee': 5, 'aaanews': 6,
                            'aaannnddd': 7, 'aaanortheast': 8, 'aabutan': 9,
                            'aacopd': 10, 'aacounty': 11, 'aacountygovt': 12,
                            'aadeshrawal': 13, 'aadya': 14, 'aadyasitara': 1
5,
                            'aafp': 16, 'aahealth': 17, 'aahh': 18, 'aai': 1
9,
                            'aaisp': 20, 'aajeevika': 21, 'aajtak': 22,
                            'aakash': 23, 'aalonzowatt': 24, 'aalto': 25,
                            'aaltouniversity': 26, 'aalwajih': 27,
                            'aamaadmi': 28, 'aamaadmiparty': 29, ...})
Модель для классификации - LogisticRegression(C=3.0, max_iter=1000)
Accuracy = 0.7047998441115858
============================
Векторизация - TfidfVectorizer(vocabulary={'aa': 0, 'aaa': 1, 'aaaaakubosan':
2, 'aaaaas': 3,
```

```
                                'aaaand': 4, 'aaachatterjee': 5, 'aaanews': 6,
                                'aaannnddd': 7, 'aaanortheast': 8, 'aabutan': 9,
                                'aacopd': 10, 'aacounty': 11, 'aacountygovt': 12,
                                'aadeshrawal': 13, 'aadya': 14, 'aadyasitara': 1
5,
                                'aafp': 16, 'aahealth': 17, 'aahh': 18, 'aai': 1
9,
                                'aaisp': 20, 'aajeevika': 21, 'aajtak': 22,
                                'aakash': 23, 'aalonzowatt': 24, 'aalto': 25,
                                'aaltouniversity': 26, 'aalwajih': 27,
                                'aamaadmi': 28, 'aamaadmiparty': 29, ...})
Модель для классификации - LinearSVC()
Accuracy = 0.7113996042675653
```

Лучший результат покаазала модель LogisticRegression(C=3.0, max_iter=1000) с CountVectorizer

word2vec

In [20]:
```
import gensim
from gensim.models import word2vec
```

In [21]:
```
from nltk import WordPunctTokenizer
from nltk.corpus import stopwords
# Подготовим корпус
corpus = []
stop_words = stopwords.words('english')
tok = WordPunctTokenizer()
for line in vocab_list:
    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 [24]:
```
corpus[1]
```

Out[24]:
```
['advice',
 'talk',
 'neighbours',
 'family',
 'exchange',
 'phone',
 'numbers',
 'create',
 'contact',
 'list',
 'phone',
 'numbers',
 'neighbours',
 'schools',
 'employer',
 'chemist',
 'gp',
```

```
 'set',
 'online',
 'shopping',
 'accounts',
 'poss',
 'adequate',
 'supplies',
 'regular',
 'meds',
 'order']
```

In [26]:
```
%time model = word2vec.Word2Vec(corpus, workers=4, min_count=10, window=10, s
```

```
CPU times: user 11.6 s, sys: 71.3 ms, total: 11.7 s
Wall time: 4.01 s
```

In [27]:
```
# Проверим, что модель обучилась
print(model.wv.most_similar(positive=['find'], topn=5))
```

```
[('try', 0.779373049736023), ('looking', 0.7359310388565063), ('gift', 0.7253
180742263794), ('delivered', 0.7166185975074768), ('meal', 0.711269140243530
3)]
```

In [ ]:
```
vectorizers_list = [CountVectorizer(vocabulary = corpusVocab), TfidfVectorize
classifiers_list = [RandomForestClassifier(), ComplementNB(), LogisticRegress
VectorizeAndClassify(vectorizers_list, classifiers_list)
```

In [38]:
```
len(corpus)
```

Out[38]:
```
44955
```

In [55]:
```
x_train.value.values[0]
```

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
<ipython-input-55-bc32498b17da> in <module>
----> 1 x_train.value.values[0]

/opt/conda/lib/python3.7/site-packages/pandas/core/generic.py in __getattr__
(self, name)
   5463             if self._info_axis._can_hold_identifiers_and_holds_name(n
ame):
   5464                 return self[name]
-> 5465             return object.__getattribute__(self, name)
   5466
   5467     def __setattr__(self, name: str, value) -> None:

AttributeError: 'Series' object has no attribute 'value'
```

In [60]:
```
boundary = 30000
X_train = corpus[:boundary]
X_test = corpus[boundary:boundary+1000]
Y_train = y_train[:boundary]
Y_test = y_train[boundary:boundary+1000]

def sentiment(v, c):
    for v in vectorizers_list:
        for c in classifiers_list:
            model = Pipeline([("vectorizer", v), ("classifier", c)])
            model.fit(X_train, Y_train)
            y_pred = model.predict(X_test)

            print('Модель для классификации - {}'.format(c))

            print_accuracy_score_for_classes(Y_test, y_pred)

            print('============================')
```

In [34]:
```
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 [61]:
```
vectorizers_list = [EmbeddingVectorizer(model.wv)]
classifiers_list = [RandomForestClassifier(),  LogisticRegression(C=3.0, solv
sentiment(vectorizers_list, classifiers_list)
```

```
Модель для классификации - RandomForestClassifier()
Метка    Accuracy
0        0.6203208556149733
1        0.3285024154589372
2        0.6610978520286396
============================
Модель для классификации - LogisticRegression(C=3.0, max_iter=1000)
Метка    Accuracy
0        0.6283422459893048
1        0.34782608695652173
2        0.7303102625298329
============================
Модель для классификации - LinearSVC()
Метка    Accuracy
0        0.6310160427807486
```

```
1       0.3140096618357488
2       0.7446300715990454
-------------------------
/opt/conda/lib/python3.7/site-packages/sklearn/svm/_base.py:986: ConvergenceW
arning: Liblinear failed to converge, increase the number of iterations.
  "the number of iterations.", ConvergenceWarning)
```

In [ ]: