Министерство науки и высшего образования Российской Федерации Федеральное государственное бюджетное образовательное учреждение высшего образования «Московский государственный технический университет имени Н.Э. Баумана (национальный исследовательский университет)» (МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ Информатика и системы управления и искусственный интеллект

КАФЕДРА Системы обработки информации и управления

Лабораторная работа №9
По курсу
«Методы машинного обучения в АСОИУ»

«Классификация текста»

Выполнил:

ИУ5-22М Киричков Е. Е.

29.05.2024

Проверил:

Балашов А.М.

Москва, 2024

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.

Ввод [1]:
```
import numpy as np
import pandas as pd
import gensim
from sklearn.feature_extraction.text import CountVectorizer, TfidfVectorizer
from sklearn.ensemble import RandomForestClassifier, GradientBoostingClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC, LinearSVC
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score
import time
import gensim.downloader as api
from tqdm import tqdm
import sys
```

Ввод [2]:
```
# Загрузка данных
train_data = pd.read_csv('data/Corona_NLP_train.csv', encoding='latin1')
test_data = pd.read_csv('data/Corona_NLP_test.csv', encoding='latin1')
```

Ввод [3]:
```
train_data.shape
```

Out[3]:
```
(41157, 6)
```

Ввод [4]:
```
train_data.head()
```

Out[4]:

	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

Ввод [5]:
```
test_data.shape
```

Out[5]:
```
(3798, 6)
```

Ввод [6]:
```
test_data.head()
```

Out[6]:

	UserName	ScreenName	Location	TweetAt	OriginalTweet	Sentiment
0	1	44953	NYC	02-03-2020	TRENDING: New Yorkers encounter empty supermar...	Extremely Negative
1	2	44954	Seattle, WA	02-03-2020	When I couldn't find hand sanitizer at Fred Me...	Positive
2	3	44955	NaN	02-03-2020	Find out how you can protect yourself and love...	Extremely Positive
3	4	44956	Chicagoland	02-03-2020	#Panic buying hits #NewYork City as anxious sh...	Negative
4	5	44957	Melbourne, Victoria	03-03-2020	#toiletpaper #dunnypaper #coronavirus #coronav...	Neutral

Ввод [7]:
```
X_train = train_data['OriginalTweet']
y_train = train_data['Sentiment']

X_test = test_data['OriginalTweet']
y_test = test_data['Sentiment']
```

Ввод [8]:
```
def check_missing(data, name):
    missing = data.isnull().sum()
    print(f'У {name} {missing} пропущенных строк')
```

Ввод [9]:
```
check_missing(train_data, 'train_data')
check_missing(test_data, 'test_data')
check_missing(X_train, 'X_train')
check_missing(X_test, 'X_test')
check_missing(y_train, 'y_train')
check_missing(y_test, 'y_test')
```

```
У train_data UserName            0
ScreenName          0
Location         8590
TweetAt             0
OriginalTweet       0
Sentiment           0
dtype: int64 пропущенных строк
У test_data UserName           0
ScreenName         0
Location         834
TweetAt            0
OriginalTweet      0
Sentiment          0
dtype: int64 пропущенных строк
У X_train 0 пропущенных строк
У X_test 0 пропущенных строк
У y_train 0 пропущенных строк
У y_test 0 пропущенных строк
```

Ввод [10]:
```
# Векторизация с помощью CountVectorizer
count_vect = CountVectorizer()
X_train_counts = count_vect.fit_transform(X_train)
X_test_counts = count_vect.transform(X_test)

# Векторизация с помощью TfidfVectorizer
tfidf_vect = TfidfVectorizer()
X_train_tfidf = tfidf_vect.fit_transform(X_train)
X_test_tfidf = tfidf_vect.transform(X_test)
```

Ввод [11]:
```
# Загрузка предобученных моделей
word2vec_model = api.load('word2vec-google-news-300')
glove_model = api.load('glove-twitter-200')
fasttext_model = api.load('fasttext-wiki-news-subwords-300')
```

Ввод [12]:
```
# Функция для усреднения векторов слов в тексте
def vectorize_text(text, model):
    words = text.split()
    vectors = [model[word] for word in words if word in model]
    if len(vectors) == 0:
        return np.zeros(model.vector_size)
    return np.mean(vectors, axis=0)

# Векторизация данных
def vectorize_dataset(dataset, model, desc="Vectorizing"):
    return np.array([vectorize_text(text, model) for text in tqdm(dataset, desc=desc)]
```

Ввод [13]:
```
X_train_w2v = vectorize_dataset(X_train, word2vec_model, desc="Vectorizing word2vec")
X_test_w2v = vectorize_dataset(X_test, word2vec_model, desc="Vectorizing word2vec")

X_train_glove = vectorize_dataset(X_train, glove_model, desc="Vectorizing Glove")
X_test_glove = vectorize_dataset(X_test, glove_model, desc="Vectorizing Glove")

X_train_fasttext = vectorize_dataset(X_train, fasttext_model, desc="Vectorizing fastTe
X_test_fasttext = vectorize_dataset(X_test, fasttext_model, desc="Vectorizing fastTex
```

```
Vectorizing word2vec: 100%|██████████| 41157/41157 [00:01<00:00, 30472.68it/s]
Vectorizing word2vec: 100%|██████████| 3798/3798 [00:00<00:00, 31254.35it/s]
Vectorizing Glove: 100%|██████████| 41157/41157 [00:01<00:00, 34204.91it/s]
Vectorizing Glove: 100%|██████████| 3798/3798 [00:00<00:00, 33523.72it/s]
Vectorizing fastText: 100%|██████████| 41157/41157 [00:01<00:00, 28920.52it/s]
Vectorizing fastText: 100%|██████████| 3798/3798 [00:00<00:00, 29443.94it/s]
```

Ввод [14]:
```
# Функции для оценки точности для каждой метки
def accuracy_score_for_classes(y_true, y_pred):
    d = {'t': y_true, 'p': y_pred}
    df = pd.DataFrame(data=d)
    classes = np.unique(y_true)
    res = dict()
    for c in classes:
        temp_data_flt = df[df['t'] == c]
        temp_acc = accuracy_score(temp_data_flt['t'].values, temp_data_flt['p'].values
        res[c] = temp_acc
    return res
```

Ввод [15]:
```
def print_accuracy_score_for_classes(y_true, y_pred):
    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]))
```

Ввод [16]:
```
# Оценка моделей
def evaluate_model(vectorizer_name, vectorizer_train, vectorizer_test, model, model_na
    start_time = time.time()
    obj_model = model
    obj_model.fit(vectorizer_train, y_train)
    predictions = obj_model.predict(vectorizer_test)

    accuracy = accuracy_score(y_test, predictions)
    duration = (time.time() - start_time) / 60

    print(f'Точность: {accuracy:.4f}, время обучения классификатора: {duration:.2f} мі
    print_accuracy_score_for_classes(y_test, predictions)
```

Ввод [17]:
```
classifiers = {
    "RandomForestClassifier": RandomForestClassifier(),
    "LinearSVC": LinearSVC(max_iter=10000),
    "LogisticRegression": LogisticRegression(max_iter=10000)
}

vectorizers = {
    "CountVectorizer": (X_train_counts, X_test_counts),
    "TfidfVectorizer": (X_train_tfidf, X_test_tfidf),
    "word2vec": (X_train_w2v, X_test_w2v),
    "glove": (X_train_glove, X_test_glove),
    "fastText": (X_train_fasttext, X_test_fasttext)
}
```

Ввод [18]:
```
for vec_name, (train_vec, test_vec) in vectorizers.items():
    for clf_name, clf in classifiers.items():
        evaluate_model(vec_name, train_vec, test_vec, clf, clf_name)
```

```
Точность: 0.4513, время обучения классификатора: 1.26 мин. (CountVectorizer + Random
ForestClassifier)
Метка    Accuracy
Extremely Negative       0.20270270270271
Extremely Positive       0.24874791318864775
Negative         0.46397694524495675
Neutral          0.6252019386106623
Positive         0.6071805702217529
Точность: 0.5284, время обучения классификатора: 0.32 мин. (CountVectorizer + Linear
SVC)
Метка    Accuracy
Extremely Negative       0.589527027027027
Extremely Positive       0.6260434056761269
Negative         0.43419788664745435
Neutral          0.6348949919224556
Positive         0.4625131995776135
Точность: 0.6087, время обучения классификатора: 0.49 мин. (CountVectorizer + Logist
icRegression)
Метка    Accuracy
Extremely Negative       0.5472972972972973
Extremely Positive       0.6126878130217028
Negative         0.5629202689721422
Neutral          0.715670436187399
Positive         0.6251319957761352
Точность: 0.4379, время обучения классификатора: 1.09 мин. (TfidfVectorizer + Random
ForestClassifier)
Метка    Accuracy
Extremely Negative       0.25844594594594594
Extremely Positive       0.2637729549248748
Negative         0.43611911623439004
Neutral          0.6009693053311793
Positive         0.5554382259767687
Точность: 0.5592, время обучения классификатора: 0.02 мин. (TfidfVectorizer + Linear
SVC)
Метка    Accuracy
Extremely Negative       0.6182432432432432
Extremely Positive       0.6460767946577629
Negative         0.4697406340057637
Neutral          0.6284329563812601
Positive         0.5205913410770855
Точность: 0.5658, время обучения классификатора: 0.20 мин. (TfidfVectorizer + Logist
icRegression)
Метка    Accuracy
Extremely Negative       0.4814189189189189
Extremely Positive       0.5208681135225376
Negative         0.5379442843419788
Neutral          0.6429725363489499
Positive         0.6272439281942978
Точность: 0.3855, время обучения классификатора: 0.95 мин. (word2vec + RandomForestC
lassifier)
Метка    Accuracy
Extremely Negative       0.22128378378378377
Extremely Positive       0.2888146911519199
Negative         0.3871277617675312
Neutral          0.3651050080775444
Positive         0.5607180570221753
Точность: 0.4292, время обучения классификатора: 0.78 мин. (word2vec + LinearSVC)
Метка    Accuracy
Extremely Negative       0.3547297297297297
Extremely Positive       0.4040066777963272
Negative         0.3919308357348703
Neutral          0.568659127625202
Positive         0.44139387539598735
Точность: 0.4371, время обучения классификатора: 0.15 мин. (word2vec + LogisticRegre
ssion)
Метка    Accuracy
Extremely Negative       0.28209459459459457
Extremely Positive       0.32387312186978295
Negative         0.4668587896253602
Neutral          0.5153473344103393
```

```
Positive         0.5216473072861668
Точность: 0.3576, время обучения классификатора: 0.81 мин. (glove + RandomForestClas
sifier)
Метка    Accuracy
Extremely Negative       0.17567567567567569
Extremely Positive       0.22370617696160267
Negative         0.37175792507204614
Neutral          0.37964458804523427
Positive         0.5258711721224921
Точность: 0.3902, время обучения классификатора: 4.27 мин. (glove + LinearSVC)
Метка    Accuracy
Extremely Negative       0.23648648648648649
Extremely Positive       0.2888146911519199
Negative         0.41786743515850144
Neutral          0.4620355411954766
Positive         0.4730728616684266
Точность: 0.4034, время обучения классификатора: 0.21 мин. (glove + LogisticRegressi
on)
Метка    Accuracy
Extremely Negative       0.24493243243243243
Extremely Positive       0.2654424040066778
Negative         0.4447646493756004
Neutral          0.44749596122778673
Positive         0.515311510031679
Точность: 0.3768, время обучения классификатора: 0.98 мин. (fastText + RandomForestC
lassifier)
Метка    Accuracy
Extremely Negative       0.22466216216216217
Extremely Positive       0.2387312186978297
Negative         0.3611911623439001
Neutral          0.42487883683360256
Positive         0.5448785638859557
Точность: 0.4323, время обучения классификатора: 0.34 мин. (fastText + LinearSVC)
Метка    Accuracy
Extremely Negative       0.3310810810810811
Extremely Positive       0.3555926544240401
Negative         0.4111431316042267
Neutral          0.5573505654281099
Positive         0.4857444561774023
Точность: 0.4160, время обучения классификатора: 0.19 мин. (fastText + LogisticRegre
ssion)
Метка    Accuracy
Extremely Negative       0.19087837837837837
Extremely Positive       0.2003338898163606
Negative         0.48222862632084534
Neutral          0.4588045234248788
Positive         0.5923970432946146
```

Итог

Наилучший результат получен с использованием CountVectorizer и LogisticRegression.

Ввод [ ]: