Министерство науки и высшего образования
Российской Федерации
Федеральное государственное бюджетное образовательное
учреждение высшего образования «Рыбинский государственный
авиационный технический университет имени П. А. Соловьева»

ИНФОРМАЦИОННЫЕ ТЕХНОЛОГИИ И СИСТЕМЫ УПРАВЛЕНИЯ

Математического и программного обеспечения электронных
вычислительных средств

ОТЧЁТ

по дисциплине:
«Методы и алгоритмы анализа данных»
на тему:
«Классификация текстов»

Выполнил: студент группы ИВМ-24 Морозов А. А.
Руководитель: ассистент Вязниковцев Д. А.

Рыбинск 2024

Содержание

Цель работы 3

1 Первичная обработка данных 4

2 Удаление стоп-слов и стемминг 5

3 «Мешок слов» и обучение моделей 7

Вывод 11

Цель работы

Целями данной лабораторной работы являются:

- изучение теории о мешке слов;
- первичная обработка данных;
- ознакомительная работа с библиотекой *nltk*;
- обучение моделей *Random Forest* и *Gradient Boosting*.

1 Первичная обработка данных

В начале необходимо загрузить 2 датасета с негативными и позитивными комментариями из популярной социальной сети:

```
import pandas as pd
dfN = pd.read_csv('negative.csv',  header=None, sep=';', quotechar="")
dfP = pd.read_csv('positive.csv',  header=None, sep=';', quotechar="")
```

Теперь конкатенируем оба *dataframe*, перемешиваем данные, добавляем имена для столбцов и выводим в консоль получившийся результат (рисунок 1.1):

```
df = pd.concat([dfN, dfP])
df = df.sample(frac=1).reset_index(drop=True)  # перемешали данные
print(f"df:\n{df}")
df.columns = ["id", "date", "name", "message", "sentiment"] +
["undefined" for i in range(df.shape[1] - 5)]
print(f"df:\n{df}")
```

```
df:
                     id        date             name                                            message  sentiment  undefined  undefined  undefined  unde
0       418668785139056641  1388653391         mirineqq  жесть, во сколько бы я не легла встаю около 11...         -1          0          0          0
1       409334773052882944  1386427989            pmzp_  RT @as_aleksandra: Кто сегодня "в стельку" отз...          1          0          1          0
2       422796605159669760  1389637540      New_Siberia        @onlinenskkpru хана, останемся без урожая:(         -1          0          0          0
3       424580121627197440  1390062763       dasha1613  RT @Pit_dO_Ob: @dasha1613 с хуя она уже такая ...         -1          0          1          0
4       411006821701525504  1386826637      vyfymyjedos  @usb_orhan, Не сплю вторую ночь... Подпольная ...          1          0          0          0
...                    ...         ...              ...                                                ...        ...        ...        ...        ...
226829  409061149142175744  1386362752      klimovich13  @kuleshikanya кстати да, удалиться это сложно ...          1          0          0          0
226830  410012637410623488  1386589605    dasha_dubovik  @pechenya2 мне Домашка предложила встречаться ...          1          0          0          0
226831  410290948950605824  1386655959     deti_i_veshi  Soda Slim для похудения. Насыпаешь в ванну и х...          1          0          0          0
226832  410835005016133632  1386785672      vhromenkova  RT @tutanuta: @vhromenkova @GigelMary @martin_...          1          0          1          0
226833  410461986338910209  1386696738  nosferatu666666  Обозреватееель женских .....))))))))))@kononyk...          1          0          0          0

[226834 rows x 12 columns]
```

Рисунок 1.1 – Получившийся *dataframe*

2 Удаление стоп-слов и стемминг

Для дальнейшей работы с *dataframe* необходимо удалить стоп-слова и выполнить стемминг оставшихся.

Для удаления стоп-слов воспользуемся функцией *preprocess_text*:

```
def preprocess_text(texts):
    stop_words = set(stopwords.words('russian'))
    regex = re.compile('[^а-я А-Я]')
    preprocess_texts = []
    for i in  tqdm.tqdm(range(len(texts))):
        text = texts[i].lower()
        text = regex.sub(' ', text)
        word_tokens = word_tokenize(text, language="russian")
        filtered_sentence = [w for w in word_tokens if not w in stop_words]
        preprocess_texts.append(' '.join(filtered_sentence))
    return preprocess_texts
```

Для стемминга воспользуемся функцией *stemming_texts*:

```
def stemming_texts(texts):
    st = SnowballStemmer("russian")
    stem_text = []
    for text in tqdm.tqdm(texts):
        word_tokens = word_tokenize(text, language="russian")
        stem_text.append(' '.join([st.stem(word) for word in word_tokens]))
    return stem_text
```

Выведем полученный *dataframe* в консоль (рисунок 2.1).

	id	date	name	message	sentiment	undefined	undefined	undefined	undefined	undefined	undefined	undefined
0	411857551764901888	1387029466	footba11master	похож шейх реш опуст неб земл	-1	0	0	0	39328	2634	607	72
1	410744049935863808	1386763987	ecewuwupor	смертельн номер смен адрес вообщ сервер хочет ...	1	0	0	0	207	208	175	0
2	413179965921054720	1387344755	NatalWinchester	хож леч поликлиник физ лечен позвоночник равн бол	-1	0	0	0	61	8	9	1
3	410806058584899584	1386778771	nononoobs	доху понрав	1	0	0	0	12889	136	60	0
4	413305514438586368	1387374211	mira_vlasyk	ког теб хотел поскор встрет мир очен хоч увидет	-1	0	1	0	6592	112	103	0
...	...	...	...	...	...	...	...	...	...	...	...	...
226829	410016715162910720	1386590577	pervert1111	седн послуша нога чет поня ошиб выбор музык	1	0	0	0	4059	216	164	0
226830	410641356755722241	1386739503	_bullsfan_	вс свалива пойд по шк пойд хоч уход	-1	0	0	0	2392	264	153	1
226831	411079149046857728	1386843881	oh_MarieMarie	ве ден компан малыш хоч дела хоч никуд идт нам...	1	0	0	0	12555	53	28	0
226832	410293488174247936	1386656565	DashaMix	добр времен суток пидар	1	0	84	0	4337	48	52	1
226833	409886762304303104	1386559594	masha_vada	бож мужчин	1	0	15	0	1920	77	93	0

Рисунок 2.1 – Изменённый *dataframe*

3 «Мешок слов» и обучение моделей

Перед обучением моделей необходимо представить текст как вектор – «Мешок слов» и его взвешенную версию с помощью функций *prepare_data* и *prepare_data_TF* соответственно:

```
def prepare_data(df, text_column, target_column, test_size=0.3, random_state=42):
    vectorizer = CountVectorizer()
    X = vectorizer.fit_transform(df[text_column])
    y = df[target_column]
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)
    return X_train, X_test, y_train, y_test

def prepare_data_TF(df, text_column, target_column, test_size=0.3, random_state=42):
    from sklearn.feature_extraction.text import TfidfVectorizer
    vectorizer = TfidfVectorizer()
    X = vectorizer.fit_transform(df[text_column])
    y = df[target_column]
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)
    return X_train, X_test, y_train, y_test
```

Для обучения модели градиентного спуска и модели случайного дерева воспользуемся функцией *learning*:

```
def learning(X_train, y_train, X_test, y_test, num, bag):
    from sklearn.model_selection import train_test_split, GridSearchCV
```

```
    from sklearn.ensemble import RandomForestClassifier,
GradientBoostingClassifier
    from sklearn.feature_extraction.text import TfidfVectorizer
    from sklearn.metrics import accuracy_score
    RFC, GBC = RandomForestClassifier(), GradientBoostingClassifier()
    param_grid_rf = {'max_depth': [5, 6, 7], 'max_features': ['sqrt', 'log2']}
    grid_rf = GridSearchCV(RFC, param_grid_rf, cv=3, n_jobs=-1)
    grid_rf.fit(X_train, y_train)
    y_predicted = grid_rf.predict(X_test)
    accuracy_score_n = accuracy_score(y_predicted, y_test)
    print(f"Лучшие параметры у дерева: {grid_rf.best_params_}, {num},
{bag}, {accuracy_score_n}")

    param_grid_gb = {'max_depth': [5, 6, 7], 'max_features': ['sqrt', 'log2']}
    grid_gb = GridSearchCV(GBC, param_grid_gb, cv=3, n_jobs=-1)
    grid_gb.fit(X_train, y_train)
    y_predicted = grid_gb.predict(X_test)
    accuracy_score_n = accuracy_score(y_predicted, y_test)
    print(f"Лучшие параметры у градиентного спуска:
{grid_gb.best_params_}, {num}, {bag}, {accuracy_score_n}")
```

Используем разные данные для обучения моделей:

- данные до удаления стоп-слов и стемминга;
- данные после удаления стоп слов, но без стемминга;
- данные после удаления стоп-слов и стемминга.

Обучение для разных данных будет производиться в нескольких потоках и результаты сохраняться в файл *output.txt*:

```
    logs = "output.txt"
    with open(logs, "w", encoding="utf-8") as file:
```

```
# Перенаправляем вывод
sys.stdout = file
X_train, X_test, y_train, y_test = prepare_data(df, "message", "sentiment")
X_train_preprocess, X_test_preprocess, y_train_preprocess, y_test_preprocess = prepare_data(preprocess_data, "message", "sentiment")
X_train_stemming, X_test_stemming, y_train_stemming, y_test_stemming = prepare_data(data_stemming, "message", "sentiment")
threading.Thread(target=learning, args=(X_train, y_train, X_test, y_test, "исходные тексты", "Обычный мешок слов")).run()
threading.Thread(target=learning, args=(X_train_preprocess, y_train_preprocess, X_test_preprocess, y_test_preprocess, "предварительно обработанные тексты", "Обычный мешок слов")).run()
threading.Thread(target=learning, args=(X_train_stemming, y_train_stemming, X_test_stemming, y_test_stemming, "тексты после стемминга", "Обычный мешок слов")).run()

X_trainTF, X_testTF, y_trainTF, y_testTF = prepare_data_TF(df, "message", "sentiment")
X_train_preprocessTF, X_test_preprocessTF, y_train_preprocessTF, y_test_preprocessTF = prepare_data_TF(preprocess_data, "message", "sentiment")
X_train_stemmingTF, X_test_stemmingTF, y_train_stemmingTF, y_test_stemmingTF = prepare_data_TF(data_stemming, "message", "sentiment")
threading.Thread(target=learning, args=(X_trainTF, y_trainTF, X_testTF, y_testTF, "исходные тексты", "Взвешенный мешок слов")).run()
threading.Thread(target=learning, args=(X_train_preprocessTF, y_train_preprocessTF, X_test_preprocessTF, y_test_preprocessTF, "предварительно обработанные тексты", "Взвешенный мешок слов")).run()
```

threading.Thread(target=learning, args=(X_train_stemmingTF, y_train_stemmingTF, X_test_stemmingTF, y_test_stemmingTF, "тексты после стемминга", "Взвешенный мешок слов")).run()

Содержимое файла *output.txt* представлено на рисунке 3.1.

```
Лучшие параметры у дерева: {'max_depth': 7, 'max_features': 'sqrt'}, исходные тексты, Обычный мешок слов, 0.581622606574481
Лучшие параметры у градиентного спуска: {'max_depth': 7, 'max_features': 'sqrt'}, исходные тексты, Обычный мешок слов, 0.6602695037545371
Лучшие параметры у дерева: {'max_depth': 7, 'max_features': 'sqrt'}, предварительно обработанные тексты, Обычный мешок слов, 0.5840178689512278
Лучшие параметры у градиентного спуска: {'max_depth': 7, 'max_features': 'sqrt'}, предварительно обработанные тексты, Обычный мешок слов, 0.6558463505312192
Лучшие параметры у дерева: {'max_depth': 7, 'max_features': 'sqrt'}, тексты после стемминга, Обычный мешок слов, 0.5863543518831464
Лучшие параметры у градиентного спуска: {'max_depth': 7, 'max_features': 'sqrt'}, тексты после стемминга, Обычный мешок слов, 0.6574480904027862
Лучшие параметры у дерева: {'max_depth': 7, 'max_features': 'sqrt'}, исходные тексты, Взвешенный мешок слов, 0.585766557434865
Лучшие параметры у градиентного спуска: {'max_depth': 7, 'max_features': 'sqrt'}, исходные тексты, Взвешенный мешок слов, 0.66516289253648
Лучшие параметры у дерева: {'max_depth': 7, 'max_features': 'sqrt'}, предварительно обработанные тексты, Взвешенный мешок слов, 0.5835770231150167
Лучшие параметры у градиентного спуска: {'max_depth': 7, 'max_features': 'sqrt'}, предварительно обработанные тексты, Взвешенный мешок слов, 0.656169637477774
Лучшие параметры у дерева: {'max_depth': 7, 'max_features': 'sqrt'}, тексты после стемминга, Взвешенный мешок слов, 0.5896606956547296
Лучшие параметры у градиентного спуска: {'max_depth': 7, 'max_features': 'sqrt'}, тексты после стемминга, Взвешенный мешок слов, 0.6551556920544885
```

Рисунок 3.1 – Содержимое файла *output.txt*.

Вывод

Полученные результаты говорят, что обе модели имеют точность более 50%, точность у градиентного спуска выше, чем у случайного дерева. Данные результаты можно объяснить тем, что тип сообщения зависит не только от набора используемых в нём слов, но и от контекста самого сообщения.