

Липецкий государственный технический университет

Факультет автоматизации и информатики

Кафедра автоматизированных систем управления

ЛАБОРАТОРНАЯ РАБОТА №3

по дисциплине «Прикладные интеллектуальные системы и экспертные

системы»

Классификация текстовых данных

Студент

Посаднев В.В.

Группа М-ИАП-22-1

Руководитель

Кургасов В.В.

Доцент

Липецк 2022 г.

Задание кафедры

1) Загрузить выборки по варианту из лабораторной работы №2

2) Используя GridSearchCV произвести предварительную обработку данных и настройку методов классификации в соответствии с заданием, вывести оптимальные значения параметров и результаты классификации модели (полнота, точность, f1-мера и аккуратности) с данными параметрами. Настройку проводить как на данных со стеммингом, так и на данных, на которых стемминг не применялся.

3) По каждому пункту работы занести в отчет программный код и результат вывода

4) Оформить сравнительную таблицу с результатами классификации различными методами с разными настройками. Сделать выводы о наиболее подходящем методе классификации ваших данных с указанием параметров метода и описанием предварительной обработки данных.

Вариант №11

Методы: MNB, DT, KNN

Мультиномиальный Наивный Байесовский метод (MNB):

- параметр сглаживания α (параметра $\alpha = \{0,1;1;2\}$)

Дерево решений (DT):

- критерий (параметр criterion: 'gini', 'entropy')
- глубина дерева (параметр max_depth от 1 до 5 с шагом 1, далее до 100 с шагом 20)

K-ближайших соседей (KNN):

- количество ближайших соседей
- метрика (евклидова, городских кварталов)

Ход работы

Загрузим выборки по варианту из лабораторной работы №2, код для выполнения данного пункта представлен на рисунке 1.

```
1 import warnings
2 from sklearn.datasets import fetch_20newsgroups
3 warnings.simplefilter(action='ignore', category=FutureWarning)

1 categories = ['misc.forsale', 'sci.med', 'talk.religion.misc']
2 remove = ['headers', 'footers', 'quotes']
3
4 twenty_train_full = fetch_20newsgroups(subset='train', categories=categories, shuffle=True, random_state=42, remove=remove)
5 twenty_test_full = fetch_20newsgroups(subset='test', categories=categories, shuffle=True, random_state=42, remove=remove)
```

Рисунок 1 – Загрузка исходных данных в соответствии с вариантом 11 из лабораторной работы №2

Зададим параметры, которые необходимо варьировать для нахождения оптимальных для более лучшей классификации модели. Параметры для каждого метода представлены на рисунке 2.

```
stop_words = [None, 'english']
max_features_values = [100, 500, 1000, 2000, 3000, 4000, 5000]
use_idf = [True, False]
```

```
dt_first = range(1, 5, 1)
dt_second = range(5, 100, 20)

decision_tree_max_depth = [*dt_first, *dt_second]
```

```
parameters_mnb = {
    'vect__max_features': max_features_values,
    'vect__stop_words': stop_words,
    'tfidf__use_idf': use_idf,
    'clf__alpha': (0.1, 1, 2)
}

parameters_dtc = {
    'vect__max_features': max_features_values,
    'vect__stop_words': stop_words,
    'tfidf__use_idf': use_idf,
    'clf__criterion': ('gini', 'entropy'),
    'clf__max_depth': decision_tree_max_depth,
}

parameters_knc = {
    'vect__max_features': max_features_values,
    'vect__stop_words': stop_words,
    'tfidf__use_idf': use_idf,
    'clf__n_neighbors': range(1, 10),
    'clf__p': (2, 1) # 2 - Евклидово, 1 - городские кварталы
}
```

Рисунок 2 – Параметры для нахождения оптимальных значений классификации

Проведем классификацию различными методами, код для которых представлен в приложении А. После проведения обучения моделей на обучающем наборе данных рассчитаем характеристики качества классификации по каждому методу.

Качество модели наивного байесовского классификатора для данных без применения стемминга и оптимальные для неё параметры представлены на рисунке 3.

```
predicted_mnb = gscv_mnb.predict(twenty_test_full.data)
print('Наивный байесовский классификатор без стема\n')
print(classification_report(twenty_test_full.target, predicted_mnb, target_names=categories))
print(gscv_mnb.best_params_)

Наивный байесовский классификатор без стема
```

	precision	recall	f1-score	support
misc.forsale	0.95	0.94	0.95	390
sci.med	0.87	0.92	0.89	396
talk.religion.misc	0.88	0.81	0.84	251
accuracy			0.90	1037
macro avg	0.90	0.89	0.90	1037
weighted avg	0.90	0.90	0.90	1037

```
{'clf__alpha': 0.1, 'tfidf__use_idf': True, 'vect__max_features': 5000, 'vect__stop_words': 'english'}
```

Рисунок 3 – Качество модели наивного байесовского классификатора без применения стемминга

Качество модели наивного байесовского классификатора для данных с применением стемминга и оптимальные для неё параметры представлены на рисунке 4.

```
predicted_mnb_stem = gscv_mnb_stem.predict(twenty_test_full.data)
print('Наивный байесовский классификатор со стемом\n')
print(classification_report(twenty_test_full.target, predicted_mnb_stem, target_names=categories))
print(gscv_mnb_stem.best_params_)

Наивный байесовский классификатор со стемом
```

	precision	recall	f1-score	support
misc.forsale	0.89	0.94	0.92	390
sci.med	0.87	0.86	0.86	396
talk.religion.misc	0.86	0.80	0.83	251
accuracy			0.88	1037
macro avg	0.87	0.87	0.87	1037
weighted avg	0.87	0.88	0.87	1037

```
{'clf__alpha': 0.1, 'tfidf__use_idf': True, 'vect__max_features': 5000, 'vect__stop_words': 'english'}
```

Рисунок 4 – Качество модели наивного байесовского классификатора с применением стемминга

Качество модели дерева решений для данных без применения стемминга и оптимальные для неё параметры представлены на рисунке 5.

```
predicted_dt = gscv_dt.predict(twenty_test_full.data)
print('Дерево решений без стема\n')
print(classification_report(twenty_test_full.target, predicted_dt, target_names=categories))
print(gscv_dt.best_params_)
```

Дерево решений без стема

	precision	recall	f1-score	support
misc.forsale	0.83	0.84	0.84	390
sci.med	0.79	0.57	0.66	396
talk.religion.misc	0.54	0.77	0.64	251
accuracy			0.72	1037
macro avg	0.72	0.73	0.71	1037
weighted avg	0.75	0.72	0.72	1037

```
{'clf__criterion': 'gini', 'clf__max_depth': 65, 'tfidf__use_idf': True, 'vect__max_features': 1000, 'vect__stop_words': 'english'}
```

Рисунок 5 – Качество модели дерева решений без применения стемминга

Качество модели дерева решений для данных после применения стемминга и оптимальные для неё параметры представлены на рисунке 6.

```
predicted_dt_stem = gscv_dt_stem.predict(twenty_test_full.data)
print('Дерево решений со стемом\n')
print(classification_report(twenty_test_full.target, predicted_dt_stem, target_names=categories))
print(gscv_dt_stem.best_params_)
```

Дерево решений со стемом

	precision	recall	f1-score	support
misc.forsale	0.80	0.72	0.76	390
sci.med	0.73	0.45	0.56	396
talk.religion.misc	0.44	0.78	0.57	251
accuracy			0.63	1037
macro avg	0.66	0.65	0.63	1037
weighted avg	0.69	0.63	0.64	1037

```
{'clf__criterion': 'gini', 'clf__max_depth': 65, 'tfidf__use_idf': False, 'vect__max_features': 2000, 'vect__stop_words': 'english'}
```

Рисунок 6 – Качество модели дерева решений после применения стемминга

Качество модели К-ближайших соседей для данных без применения стемминга и оптимальные для неё параметры представлены на рисунке 7.

```
predicted_knc = gscv_knc.predict(twenty_test_full.data)
print('К-ближайших соседей без стема\n')
print(classification_report(twenty_test_full.target, predicted_knc, target_names=categories))
print(gscv_knc.best_params_)
```

К-ближайших соседей без стема

	precision	recall	f1-score	support
misc.forsale	0.77	0.81	0.79	390
sci.med	0.71	0.61	0.66	396
talk.religion.misc	0.53	0.61	0.57	251
accuracy			0.69	1037
macro avg	0.67	0.68	0.67	1037
weighted avg	0.69	0.69	0.69	1037

```
{'clf__n_neighbors': 4, 'clf__p': 2, 'tfidf__use_idf': True, 'vect__max_features': 100, 'vect__stop_words': 'english'}
```

Рисунок 7 – Качество модели К-ближайших соседей без применения стемминга

Качество модели К-ближайших соседей для данных после применения стемминга и оптимальные для неё параметры представлены на рисунке 8.

```
predicted_knc_stem = gscv_knc_stem.predict(twenty_test_full.data)
print('К-ближайших соседей со стемом\n')
print(classification_report(twenty_test_full.target, predicted_knc_stem, target_names=categories))
print(gscv_knc_stem.best_params_)
```

К-ближайших соседей со стемом

	precision	recall	f1-score	support
misc.forsale	0.62	0.80	0.70	390
sci.med	0.58	0.61	0.59	396
talk.religion.misc	0.71	0.33	0.45	251
accuracy			0.61	1037
macro avg	0.64	0.58	0.58	1037
weighted avg	0.63	0.61	0.60	1037

```
{'clf__n_neighbors': 2, 'clf__p': 2, 'tfidf__use_idf': True, 'vect__max_features': 100, 'vect__stop_words': 'english'}
```

Рисунок 8 – Качество модели К-ближайших соседей после применения стемминга

Заключение

В ходе выполнения данной лабораторной работы мною были получены навыки классификации текстовых данных. В рамках данной работы были применены различные методы классификации: наивный байесовский классификатор, дерево решений и К-ближайших соседей.

Наилучшей классификацией для набора данных обладает модель наивного байесовского классификатора без использования стемминга. Её точность составляет 90%. Параметры для данной модели также представлены на рисунке 3.

Приложение А

Исходный код

```
#!/usr/bin/env python
# coding: utf-8

# # Выгрузка данных из лабораторной работы №2 (вариант №11)
# ## Вариант №11: [misc.forsale, sci.med, talk.religion.misc]

# In[1]:

import warnings
from sklearn.datasets import fetch_20newsgroups
warnings.simplefilter(action='ignore', category=FutureWarning)

# In[2]:

categories = ['misc.forsale', 'sci.med', 'talk.religion.misc']
remove = ['headers', 'footers', 'quotes']

twenty_train_full = fetch_20newsgroups(subset='train', categories=categories,
shuffle=True, random_state=42, remove=remove)
twenty_test_full = fetch_20newsgroups(subset='test', categories=categories,
shuffle=True, random_state=42, remove=remove)

# # Применение стемминга

# In[3]:

import nltk
from nltk import word_tokenize
from nltk.stem import *

nltk.download('punkt')

# In[4]:

def stemming(data):
    porter_stemmer = PorterStemmer()
    stem = []
    for text in data:
        nltk_tokens = word_tokenize(text)
        line = ''.join([' ' + porter_stemmer.stem(word) for word in
nltk_tokens])
        stem.append(line)
    return stem

# In[5]:

stem_train = stemming(twenty_train_full.data)
stem_test = stemming(twenty_test_full.data)
```

```

# # Вариант №11
# ## Методы: [MNB, DT, KNN]

# In[6]:

from sklearn.naive_bayes import MultinomialNB
from sklearn.tree import DecisionTreeClassifier
from sklearn.neighbors import KNeighborsClassifier

# In[7]:

stop_words = [None, 'english']
max_features_values = [100, 500, 1000, 2000, 3000, 4000, 5000]
use_idf = [True, False]

# In[8]:

dt_first = range(1, 5, 1)
dt_second = range(5, 100, 20)

decision_tree_max_depth = [*dt_first, *dt_second]

# In[9]:

parameters_mnb = {
    'vect__max_features': max_features_values,
    'vect__stop_words': stop_words,
    'tfidf__use_idf': use_idf,
    'clf__alpha': (0.1, 1, 2)
}

parameters_dtc = {
    'vect__max_features': max_features_values,
    'vect__stop_words': stop_words,
    'tfidf__use_idf': use_idf,
    'clf__criterion': ('gini', 'entropy'),
    'clf__max_depth': decision_tree_max_depth,
}

parameters_knc = {
    'vect__max_features': max_features_values,
    'vect__stop_words': stop_words,
    'tfidf__use_idf': use_idf,
    'clf__n_neighbors': range(1, 10),
    'clf__p': (2, 1) # 2 - Евклидово, 1 - городские кварталы
}

# In[10]:

from sklearn.model_selection import GridSearchCV
from sklearn.pipeline import Pipeline
from sklearn.feature_extraction.text import CountVectorizer, TfidfTransformer

```

```

# # Наивный байесовский классификатор

# ## Без использования стема

# In[11]:

text_clf_mnb = Pipeline([('vect', CountVectorizer()),
                          ('tfidf', TfidfTransformer()),
                          ('clf', MultinomialNB())])
gscv_mnb = GridSearchCV(text_clf_mnb, param_grid=parameters_mnb, n_jobs=-1)
gscv_mnb.fit(twenty_train_full.data, twenty_train_full.target)

# ## С использованием стема

# In[12]:

text_clf_mnb_stem = Pipeline([('vect', CountVectorizer()),
                               ('tfidf', TfidfTransformer()),
                               ('clf', MultinomialNB())])
gscv_mnb_stem = GridSearchCV(text_clf_mnb_stem, param_grid=parameters_mnb,
                              n_jobs=-1)
gscv_mnb_stem.fit(stem_train, twenty_train_full.target)

# # Дерево решений

# ## Без использования стема

# In[13]:

text_clf_dt = Pipeline([('vect', CountVectorizer()),
                         ('tfidf', TfidfTransformer()),
                         ('clf', DecisionTreeClassifier())])
gscv_dt = GridSearchCV(text_clf_dt, param_grid=parameters_dtc, n_jobs=-1)
gscv_dt.fit(twenty_train_full.data, twenty_train_full.target)

# ## С использованием стема

# In[14]:

text_clf_dt_stem = Pipeline([('vect', CountVectorizer()),
                              ('tfidf', TfidfTransformer()),
                              ('clf', DecisionTreeClassifier())])
gscv_dt_stem = GridSearchCV(text_clf_dt_stem, param_grid=parameters_dtc,
                              n_jobs=-1)
gscv_dt_stem.fit(stem_train, twenty_train_full.target)

# # K-ближайших соседей

# ## Без использования стема

# In[15]:

text_clf_knc = Pipeline([('vect', CountVectorizer()),
                          ('tfidf', TfidfTransformer()),

```

```

        ('clf', KNeighborsClassifier()))
gscv_knc = GridSearchCV(text_clf_knc, param_grid=parameters_knc, n_jobs=-1)
gscv_knc.fit(twenty_train_full.data, twenty_train_full.target)

# ## С использованием стема

# In[16]:

text_clf_knc_stem = Pipeline([('vect', CountVectorizer()),
                              ('tfidf', TfidfTransformer()),
                              ('clf', KNeighborsClassifier())])
gscv_knc_stem = GridSearchCV(text_clf_knc_stem, param_grid=parameters_knc,
n_jobs=-1)
gscv_knc_stem.fit(stem_train, twenty_train_full.target)

# # Вывод полученных результатов анализа

# In[17]:

from sklearn.metrics import classification_report

# In[18]:

predicted_mnb = gscv_mnb.predict(twenty_test_full.data)
print('Наивный байесовский классификатор без стема\n')
print(classification_report(twenty_test_full.target, predicted_mnb,
target_names=categories))
print(gscv_mnb.best_params_)

# In[19]:

predicted_mnb_stem = gscv_mnb_stem.predict(twenty_test_full.data)
print('Наивный байесовский классификатор со стемом\n')
print(classification_report(twenty_test_full.target, predicted_mnb_stem,
target_names=categories))
print(gscv_mnb_stem.best_params_)

# In[20]:

predicted_dt = gscv_dt.predict(twenty_test_full.data)
print('Дерево решений без стема\n')
print(classification_report(twenty_test_full.target, predicted_dt,
target_names=categories))
print(gscv_dt.best_params_)

# In[21]:

predicted_dt_stem = gscv_dt_stem.predict(twenty_test_full.data)
print('Дерево решений со стемом\n')
print(classification_report(twenty_test_full.target, predicted_dt_stem,
target_names=categories))

```

```

print(gscv_dt_stem.best_params_)

# In[22]:

predicted_knc = gscv_knc.predict(twenty_test_full.data)
print('К-ближайших соседей без стема\n')
print(classification_report(twenty_test_full.target, predicted_knc,
target_names=categories))
print(gscv_knc.best_params_)

# In[23]:

predicted_knc_stem = gscv_knc_stem.predict(twenty_test_full.data)
print('К-ближайших соседей со стемом\n')
print(classification_report(twenty_test_full.target, predicted_knc_stem,
target_names=categories))
print(gscv_knc_stem.best_params_)

# # Сравнительная таблица

# In[24]:

import pandas as pd

# In[25]:

writer = pd.ExcelWriter('result.xlsx', engine='openpyxl')

# Наивный байесовский классификатор без стема
df1 = pd.DataFrame(classification_report(predicted_mnb,
twenty_test_full.target, output_dict=True))
# Наивный байесовский классификатор со стемом
df2 = pd.DataFrame(classification_report(predicted_mnb_stem,
twenty_test_full.target, output_dict=True))
# Дерево решений без стема
df3 = pd.DataFrame(classification_report(predicted_dt,
twenty_test_full.target, output_dict=True))
# Дерево решений со стемом
df4 = pd.DataFrame(classification_report(predicted_dt_stem,
twenty_test_full.target, output_dict=True))
# К-ближайших соседей без стема
df5 = pd.DataFrame(classification_report(predicted_knc,
twenty_test_full.target, output_dict=True))
# К-ближайших соседей со стемом
df6 = pd.DataFrame(classification_report(predicted_knc_stem,
twenty_test_full.target, output_dict=True))

df1.to_excel(writer, sheet_name='НБК без стема')
df2.to_excel(writer, sheet_name='НБК со стемом')
df3.to_excel(writer, sheet_name='Дерево решений без стема')
df4.to_excel(writer, sheet_name='Дерево решений со стемом')
df5.to_excel(writer, sheet_name='К-ближайших соседей без стема')
df6.to_excel(writer, sheet_name='К-ближайших соседей со стемом')

writer.save()

```

```
# In[26]:
```

```
gscv_mnb.best_params_
```