Липецкий государственный технический университет

Факультет автоматизации и информатики

Кафедра автоматизированных систем управления

ЛАБОРАТОРНАЯ РАБОТА №2

по дисциплине «Прикладные интеллектуальные системы и экспертные системы»

Предварительная обработка текстовых данных

Студент Мамедов Р. В.

Группа М-ИАП-23-1

Руководитель Кургасов В.В.

Доцент

Липецк 2023 г.

Цель работы

Получить практические навыки обработки текстовых данных в среде Jupiter Notebook. Научиться проводить предварительную обработку текстовых данных и выявлять параметры обработки, позволяющие добиться наилучшей точности классификации.

Задание кафедры

1) В среде Jupiter Notebook создать новый ноутбук (Notebook)

2) Импортировать необходимые для работы библиотеки и модули

3) Загрузить обучающую и экзаменационную выборку в соответствие с вариантом

4) Вывести на экран по одному-два документа каждого класса.

5) Применить стемминг, записав обработанные выборки (тестовую и обучающую) в новые переменные.

6) Провести векторизацию выборки:

a. Векторизовать обучающую и тестовую выборки простым подсчетом слов (CountVectorizer) и значеним max_features = 10000

b. Вывести и проанализировать первые 20 наиболее частотных слов всей выборки и каждого класса по-отдельности.

c. Применить процедуру отсечения стоп-слов и повторить пункт b.

d. Провести пункты a – c для обучающей и тестовой выборки, для которой проведена процедура стемминга.

e. Векторизовать выборки с помощью TfidfTransformer (с использованием TF и TF-IDF взвешиваний) и повторить пункты b-d.

7) По результатам пункта 6 заполнить таблицы наиболее частотными терминами обучающей выборки и каждого класса по отдельности. Всего должно получиться по 4 таблицы для выборки, к которой применялась операция стемминга и 4 таблицы для выборки, к которой операция стемминга не применялась.

Без стемминга						
	Count		TF		TF-IDF	
№	Без стоп-слов	С стоп-словами	Без стоп-слов	С стоп-словами	Без стоп-слов	С стоп-словами
1						
2						
…						
20						

Со стеммингом						
	Count		TF		TF-IDF	
№	Без стоп-слов	С стоп-словами	Без стоп-слов	С стоп-словами	Без стоп-слов	С стоп-словами
1						
2						
…						
20						

8) Используя конвейер (Pipeline) реализовать модель Наивного Байесовского классификатора и выявить на основе показателей качества (значения полноты, точности, f1-меры и аккуратности), какая предварительная обработка данных обеспечит наилучшие результаты классификации. Должны быть исследованы следующие характеристики:

• Наличие - отсутствие стемминга

• Отсечение – не отсечение стоп-слов

• Количество информативных терминов (max_features)

• Взвешивание: Count, TF, TF-IDF

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

10) По результатам классификации занести в отчет выводы о наиболее подходящей предварительной обработке данных (наличие стемминга, взвешивание терминов, стоп-слова, количество информативных терминов).

Вариант №12

Классы 2, 12, 13 ('comp.graphics', 'sci.crypt', 'sci.electronics')

Загрузим обучающую и экзаменационную выборку в соответствии с вариантом. Код для загрузки данных представлен на рисунке 1.

```
remove = ('headers', 'footers', 'quotes')


def get_train_data(categories):
    if type(categories) is not list:
        categories = [categories]
    return fetch_20newsgroups(subset='train', shuffle=True, categories=categories, random_state=42, remove=remove)


all_categories = ['comp.graphics', 'sci.crypt', 'sci.electronics']
train_bunch = get_train_data(all_categories)
test_bunch = fetch_20newsgroups(subset='test', shuffle=True, random_state=42, categories=all_categories, remove=remove)
```

Рисунок 1 – Код для загрузки данных

После успешной загрузки данных посмотрим на записи. Для этого выведем по одному документу каждого класса из обучающей выборки. Результаты представлены на рисунке 2.

```
get_sample(train_bunch, all_categories.index('comp.graphics'))
```
Executed at 2023.11.11 12:51:09 in 6ms

```
"Hello, I realize that this might be a FAQ but I have to ask since I don't get a\nchange to read this newsgroup very
often.  Anyways for my senior project I need\nto convert an AutoCad file to a TIFF file.  Please I don't need anyone
telling\nme that the AutoCAD file is a vector file and the TIFF is a bit map since I\nhave heard that about 100 times
 already I would just like to know if anyone\nknows how to do this or at least point me to the right direction."
```

```
get_sample(train_bunch, all_categories.index('sci.crypt'))
```
Executed at 2023.11.11 12:51:09 in 24ms

```
'Looking for PostScript or Tex version of a paper called:\n\t"PUBLIC-KEY CRYPTOGRAPHY"\n\nWritten by:\n\tJames
Nechvatal\n\tSecurity Technology Group\n\tNational Computer Systems Laboratory\n\tNational Institute of Standards and
 Technology\n\tGaithersburg, MD 20899\n\n\tDecember 1990\n\nThe version I obtained is plain text and all symbolic
character\nformatting has been lost.\n'
```

```
get_sample(train_bunch, all_categories.index('sci.electronics'))
```
Executed at 2023.11.11 12:51:09 in 54ms

```
'Just a thought........Maybe it possibly has to do with the fact that it\nIS an Emerson.  I\'ve got an Emerson VCR
which is #6 in the series.  Returned\nit six times for various and never the same problems.  Got tired of taking it
\nback and fixed it myself.  The Hi-Fi "window" was a bit off.  Something like\nthe Hi-Fi audio fine-tuning.  When I
was a Wal-Mart "associate" in \'88-\'89,\nwe had AT LEAST one returned as defective EVERY SINGLE DAY.  How\'s that
for\nreliability?  Face it--Emerson can make audio stuff (albeit not of premium\nquality), but they CAN\'T make
anything as complex as video equipment with \nreliability IMHO.  Please, no flames.  Just *had* to share my Emerson
disaster\nin the light of this exploding tv.  \nJC\n\n\n'
```

Рисунок 2 – Пример загруженных данных

Применим стемминг к исходным данным в соответствии с кодом и посмотрим на обработанные данные, которые представлены на рисунке 3.

```python
def stemminize(documents: list[str]) -> list[str]:
    porter_stemmer = PorterStemmer()
    stem_train = []
    for document in documents:
        nltk_tokens = word_tokenize(document)
        line = ''
        for word in nltk_tokens:
            line += ' ' + porter_stemmer.stem(word)
        stem_train.append(line)
    return stem_train


train_tokenized = stemminize(train_bunch.data)
test_tokenized = stemminize(test_bunch.data)
```

Executed at 2023.11.11 12:51:16 in 7s 257ms

```
[nltk_data] Downloading package punkt to
[nltk_data]     C:\Users\Ruslan\AppData\Roaming\nltk_data...
[nltk_data]   Package punkt is already up-to-date!
```

```python
# вывод 3 первых документов обучающих данных
train_tokenized[:3]
```

Executed at 2023.11.11 12:51:16 in 4ms

```
[" hello , i realiz that thi might be a faq but i have to ask sinc i do n't get a chang to read thi newsgroup veri
often . anyway for my senior project i need to convert an autocad file to a tiff file . pleas i do n't need anyon
tell me that the autocad file is a vector file and the tiff is a bit map sinc i have heard that about 100 time
alreadi i would just like to know if anyon know how to do thi or at least point me to the right direct .",
```

```python
# вывод 3 первых документов тестовых данных
test_tokenized[:3]
```

Executed at 2023.11.11 12:51:16 in 23ms

```
[' well , i am place a file at my ftp today that contain sever polygon descript of a head , face , skull , vase ,
etc . the format of the file is a list of vertic , normal , and triangl . there are variou resolut and the name of
the data file includ the number of polygon , eg . phred.1.3k.vbl contain 1300 polygon . in order to get the data via
```

Рисунок 3 – Данные, обработанные стеммингом

Проведем векторизацию выборки. Для этого векторизуем обучающую и тестовую выборку простым подсчетом слов с использованием класса CountVectorizer и значением max_features = 10000, код для выполнения данного способа представлен на рисунке 4. Выведем первые 20 наиболее частотных слов по всей выборки и отобразим на рисунке 5.

```
from sklearn.feature_extraction.text import CountVectorizer

vect = CountVectorizer(max_features=10000)
train_data = vect.fit_transform(train_bunch.data)


def get_20_freq_words(vect, data):
    words = list(zip(vect.get_feature_names_out(), np.ravel(data.sum(axis=0))))
    words.sort(key=lambda x: x[1], reverse=True)
    return words[:20]


get_20_freq_words(vect, train_data)
```

Executed at 2023.11.11 12:51:16 in 172ms

```
[('the', 16689),
 ('to', 8883),
 ('of', 7021),
 ('and', 6843),
 ('is', 5467),
 ('in', 4416),
 ('it', 3900),
 ('that', 3682),
 ('for', 3677),
 ('you', 2852),
 ('be', 2788),
 ('this', 2585),
 ('on', 2451),
 ('are', 2155),
```

Рисунок 4 – Код для векторизации обучающей выборки простым подсчетом слов

```
vect = CountVectorizer(max_features=10000)
dtm = vect.fit_transform(test_bunch.data)

get_20_freq_words(vect, dtm)
Executed at 2023.11.11 12:51:22 in 106ms
```

```
[('the', 9066),
 ('to', 5360),
 ('of', 4137),
 ('and', 4073),
 ('is', 3074),
 ('in', 2610),
 ('it', 2402),
 ('for', 2362),
 ('that', 2228),
 ('you', 2086),
 ('be', 1535),
 ('this', 1472),
 ('on', 1462),
 ('or', 1295),
```

Рисунок 5 – Результат векторизации тестовой выборки простым подсчетом слов

Применим процедуру отсечения стоп-слов и повторим вывод полученных результатов. Код для обработки данных путем отсечения стоп- 7 слов представлен на рисунке 6. Результат векторизации обучающей и тестовой выборки простым подсчетом слов с отсечением стоп-слов представлен на рисунке 7.

```
vect = CountVectorizer(max_features=10000, stop_words='english')
dtm = vect.fit_transform(train_bunch.data)

get_20_freq_words(vect, dtm)
Executed at 2023.11.11 12:51:19 in 133ms
```

```
[('key', 937),
 ('use', 932),
 ('like', 642),
 ('don', 592),
 ('db', 562),
 ('edu', 553),
 ('encryption', 552),
 ('data', 547),
 ('know', 542),
 ('just', 533),
 ('chip', 521),
 ('does', 501),
 ('used', 498),
 ('information', 497),
```

Рисунок 6 – Код для векторизации обучающей простым подсчетом слов с отсечением стоп-слов

```
vect = CountVectorizer(max_features=10000, stop_words='english')
dtm = vect.fit_transform(test_bunch.data)
get_20_freq_words(vect, dtm)
```
Executed at 2023.11.11 16:04:46 in 111ms

```
[('image', 666),
 ('jpeg', 526),
 ('use', 516),
 ('edu', 468),
 ('graphics', 462),
 ('like', 408),
 ('file', 389),
 ('don', 378),
 ('data', 368),
 ('know', 355),
 ('just', 339),
 ('bit', 337),
 ('available', 325),
 ('software', 324),
```

Рисунок 7 – Результат векторизации обучающей и тестовой выборки простым подсчетом слов с отсечением стоп-слов

Также проведем аналогичный анализ для данных после стемминга. Результат векторизации обучающей и тестовой выборки после стемминга простым подсчетом слов без отсечения стоп-слов представлен на рисунке 8. Результат векторизации обучающей и тестовой выборки после стемминга простым подсчетом слов с отсечением стоп-слов представлен на рисунке 9.

```
vect = CountVectorizer(max_features=10000)
dtm = vect.fit_transform(train_tokenized)
print(get_20_freq_words(vect, dtm))
```
Executed at 2023.11.11 16:09:41 in 159ms

```
[('the', 16688), ('to', 8883), ('of', 7021), ('and', 6843), ('is',
5549), ('in', 4419), ('it', 4191), ('that', 3692), ('for', 3677),
```

Рисунок 8 – Результат векторизации обучающей после стемминга простым подсчетом слов без отсечения стоп-слов

```
vect = CountVectorizer(max_features=10000, stop_words='english')
dtm = vect.fit_transform(train_tokenized)
print(get_20_freq_words(vect, dtm))
```
Executed at 2023.11.11 16:09:38 in 75ms

```
[('thi', 2585), ('use', 2014), ('key', 1283), ('ha', 887), ('ani',
866), ('wa', 783), ('encrypt', 774), ('imag', 737), ('file', 730),
('like', 711), ('chip', 672), ('doe', 671), ('know', 622), ('bit',
621), ('program', 569), ('db', 562), ('onli', 560), ('edu', 553),
('data', 548), ('secur', 534)]
```

Рисунок 9 – Результат векторизации обучающей выборки после стемминга простым подсчетом слов с отсечением стоп-слов

Воспользуемся векторизацией выборки с помощью TfidfTransformer (с использованием TF и TF-IDF взвешиваний). Векторизация выборки с использованием TfidfTransformer для набора данных без использования стоп-слов представлен на рисунке 10, с использованием стоп-слов представлен на рисунке 11.

```
vectorizer = CountVectorizer(max_features=10000)
dtm = vectorizer.fit_transform(train_bunch.data)
tfidf = TfidfTransformer(use_idf=True).fit_transform(dtm)

feature_names = vectorizer.get_feature_names_out()
tfidf_values = tfidf.toarray().sum(axis=0)


def get_20_freq_words_idf(feature_names, tfidf_values):
    word_weights = dict(zip(feature_names, tfidf_values))
    sorted_words = sorted(word_weights.items(), key=lambda x: x[1], reverse=True)
    for word, weight in sorted_words[:20]:
        print(f"{word}: {weight}")


get_20_freq_words_idf(feature_names, tfidf_values)
```

Executed at 2023.11.11 12:51:37 in 211ms

```
the: 214.27923011983643
to: 128.50658234239415
of: 101.86203264775209
and: 91.733961272254
is: 83.97727336110019
it: 79.47554322700573
in: 72.70953833589493
that: 72.51290798766651
for: 67.66475343892618
you: 66.41180952122423
be: 55.65313856163747
this: 55.13173979874598
on: 50.56554310512375
have: 48.11707358740201
```

Рисунок 10 – Результат векторизации набора данных без использования стоп- слов

```
vectorizer = CountVectorizer(max_features=10000, stop_words='english')
dtm = vectorizer.fit_transform(train_bunch.data)
tfidf = TfidfTransformer(use_idf=True).fit_transform(dtm)

feature_names = vectorizer.get_feature_names_out()
tfidf_values = tfidf.toarray().sum(axis=0)

get_20_freq_words_idf(feature_names, tfidf_values)
```

Executed at 2023.11.11 12:51:40 in 190ms

```
key: 31.639250020331772
know: 29.275985507770525
use: 28.5909199903978
like: 27.36974112627459
does: 27.32020401912984
don: 25.935560274661245
just: 25.519485103509275
chip: 24.453605471872685
thanks: 24.11088770306822
encryption: 20.612197002323146
good: 20.345796034175873
need: 19.637524126929815
ve: 19.573641488242785
graphics: 19.457067951393185
```

Рисунок 11 – Результат векторизации набора данных с использованием стоп- слов

Проведем аналогичную векторизацию для набора данных после стемминга. Результат векторизации набора данных после стемминга без использования стоп-слов представлен на рисунке 12, с использованием стопслов представлен на рисунке 13.

```
vectorizer = CountVectorizer(max_features=10000)
dtm = vectorizer.fit_transform(train_tokenized)
tfidf = TfidfTransformer(use_idf=True).fit_transform(dtm)

feature_names = vectorizer.get_feature_names_out()
tfidf_values = tfidf.toarray().sum(axis=0)

get_20_freq_words_idf(feature_names, tfidf_values)
```

Executed at 2023.11.11 12:51:43 in 220ms

```
the: 214.47110783764484
to: 128.91799924792693
of: 101.9397995673158
and: 91.85258731878157
is: 84.95276457618122
it: 82.95304153471035
in: 73.08110129504044
that: 73.01505717147099
for: 68.08421571530532
you: 66.66827307818957
be: 59.109398696493265
thi: 55.45567314375222
on: 50.853304634787605
have: 50.30994107364992
```

Рисунок 12 – Результат векторизации набора данных после стемминга без использования стоп-слов

```
vectorizer = CountVectorizer(max_features=10000, stop_words='english'
dtm = vectorizer.fit_transform(train_bunch.data)
tfidf = TfidfTransformer(use_idf=True).fit_transform(dtm)

feature_names = vectorizer.get_feature_names_out()
tfidf_values = tfidf.toarray().sum(axis=0)

get_20_freq_words_idf(feature_names, tfidf_values)
```

Executed at 2023.11.11 12:51:51 in 205ms

```
key: 31.639250020331772
know: 29.275985507770525
use: 28.5909199903978
like: 27.36974112627459
does: 27.32020401912984
don: 25.935560274661245
just: 25.519485103509275
chip: 24.453605471872685
thanks: 24.11088770306822
encryption: 20.612197002323146
good: 20.345796034175873
need: 19.637524126929815
ve: 19.573641488242785
graphics: 19.457067951393185
```

Рисунок 13 – Результат векторизации набора данных после стемминга с использованием стоп-слов

Составим сводную таблицу для отображения результатов векторизации и сохраним её в файл Excel. Составленная таблица для обучающего набора данных без применения стемминга представлена на рисунке 14. Для тестового набора данных без применения стемминга представлена на рисунке 15. Для обучающего набора данных с применением стемминга представлен на рисунке
16. Для тестового набора данных с применением стемминга представлен на рисунке 17.

	Count		TF		TF-IDF	
	Без стоп-слов	С стоп-словами	Без стоп-слов	С стоп-словами	Без стоп-слов	С стоп-словами
0	('the', 16689)	('key', 937)	('the', 573.587874601753	('use', 58.433522994302784)	('the', 214.27923011983643)	('key', 31.639250020331772)
1	('to', 8883)	('use', 932)	('to', 323.08392380711285	('know', 57.307906073228544)	('to', 128.50658234239415)	('know', 29.275985507770525)
2	('of', 7021)	('like', 642)	('of', 242.72988399270668	('like', 56.64353296005652)	('of', 101.86203264775209)	('use', 28.5909199903978)
3	('and', 6843)	('don', 592)	('and', 217.110392007833	('don', 49.839861337882176)	('and', 91.733961272254)	('like', 27.36974112627459)
4	('is', 5467)	('db', 562)	('is', 186.06314755308978	('just', 49.002983276208454)	('is', 83.97727336110019)	('does', 27.32020401912984)
5	('in', 4416)	('edu', 553)	('it', 169.68759255688352	('does', 48.9585366753105)	('it', 79.47554322700573)	('don', 25.935560274661245)
6	('it', 3900)	('encryption', 552	('in', 162.3235233209576)	('key', 48.142224601803)	('in', 72.70953833589493)	('just', 25.519485103509275)
7	('that', 3682)	('data', 547)	('for', 148.5456621151009	('thanks', 39.15072837325299)	('that', 72.51290798766651)	('chip', 24.453605471872685)
8	('for', 3677)	('know', 542)	('that', 146.625950865246	('chip', 35.95980854400468)	('for', 67.66475343892618)	('thanks', 24.11088770306822)
9	('you', 2852)	('just', 533)	('you', 120.107267659313	('good', 35.08471270781613)	('you', 66.41180952122423)	('encryption', 20.612197002323146)
10	('be', 2788)	('chip', 521)	('this', 105.177322997787	('need', 32.70371344296186)	('be', 55.65313856163747)	('good', 20.345796034175873)
11	('this', 2585)	('does', 501)	('be', 98.40643227700879	('used', 31.85354729678519)	('this', 55.13173979874598)	('need', 19.637524126929815)
12	('on', 2451)	('used', 498)	('on', 96.18830762929639	('think', 31.408220640281492)	('on', 50.56554310512375)	('ve', 19.573641488242785)
13	('are', 2155)	('information', 49	('have', 86.456129561767	('ve', 31.265005741849148)	('have', 48.11707358740201)	('graphics', 19.457067951393185)
14	('with', 2111)	('image', 492)	('with', 76.859879676853	('time', 30.354985689209872)	('are', 44.12242787539058)	('clipper', 19.278143372470023)
15	('or', 2090)	('people', 483)	('if', 75.80342481606581)	('people', 30.323719447911316	('with', 43.883460758799835)	('people', 18.825771055423083)
16	('have', 1879)	('time', 447)	('or', 75.78439948121508	('encryption', 28.319287928894	('if', 43.62474130570933)	('think', 18.438571829249657)
17	('as', 1784)	('bit', 437)	('are', 74.94851602350174	('using', 27.62980799728084)	('or', 43.39770046917348)	('used', 18.158442956203725)
18	('can', 1704)	('file', 427)	('can', 68.7769234754326	('graphics', 27.205639758499185	('can', 40.89942210826312)	('government', 17.929344754138906
19	('if', 1702)	('graphics', 423)	('not', 63.1163398649122	('clipper', 26.571725044843287	('as', 40.345366794020045)	('time', 17.5444284746565)

Рисунок 14 – Таблица результата векторизации для обучающего набора данных без применения стемминга

	Count		TF		TF-IDF	
	Без стоп-слов	С стоп-словами	Без стоп-слов	С стоп-словами	Без стоп-слов	С стоп-словами
0	('the', 9066)	('image', 666)	('the', 351.8833889205307)	('know', 42.06416144310403)	('the', 132.73987348944183)	('know', 21.435150574064025)
1	('to', 5360)	('jpeg', 526)	('to', 215.21048224463186)	('like', 36.822127481402006)	('to', 85.24139589140572)	('like', 18.950713675342715)
2	('of', 4137)	('use', 516)	('of', 153.66848738794562)	('use', 36.792352685034714)	('of', 64.8113206497389)	('use', 18.420212557191185)
3	('and', 4073)	('edu', 468)	('and', 140.95239513687198)	('just', 32.40544450511485)	('and', 59.482599607653924)	('thanks', 17.098531494337998)
4	('is', 3074)	('graphics', 462)	('is', 120.607868681005)	('don', 30.469003399281732)	('is', 54.52831275498656)	('does', 16.98361584501106)
5	('in', 2610)	('like', 408)	('it', 110.88904071825)	('does', 29.885181249520144)	('it', 53.75496178917013)	('just', 16.684961307647363)
6	('it', 2402)	('file', 389)	('in', 104.33536769410595)	('thanks', 27.29259689277586)	('that', 47.770429081592496)	('don', 16.428887334276713)
7	('for', 2362)	('don', 378)	('that', 98.57686193779719)	('think', 24.82104074312471)	('in', 47.06683267580104)	('think', 14.557804515797576)
8	('that', 2228)	('data', 368)	('for', 92.40323586413749)	('used', 21.459669060506144)	('you', 45.42774190604445)	('graphics', 14.252209873941238)
9	('you', 2086)	('know', 355)	('you', 81.3739413442512)	('need', 20.686196868337543)	('for', 43.184971775965785)	('program', 13.64987318610155)
10	('be', 1535)	('just', 339)	('be', 65.317483332802)	('graphics', 20.679160827517396)	('be', 35.43111523777322)	('government', 12.964530900368489)
11	('this', 1472)	('bit', 337)	('on', 62.06126483104049)	('time', 20.129142247758516)	('this', 34.08603226381701)	('chip', 12.860296724296857)
12	('on', 1462)	('available', 325)	('this', 61.59725664266683)	('program', 19.584807797820034	('on', 32.90027716297289)	('used', 12.439336643011975)
13	('or', 1295)	('software', 324)	('have', 57.61931147617616)	('people', 19.102579785729354)	('have', 32.025491313098804)	('people', 12.333813168768687)
14	('with', 1258)	('images', 307)	('or', 51.777520899187046)	('chip', 18.54157887888643)	('if', 29.12956299199104)	('need', 12.240898928564446)
15	('have', 1215)	('program', 298)	('if', 50.32542218561264)	('edu', 18.28349107306669)	('or', 29.011357783500337)	('bit', 12.015183440146776)
16	('are', 1186)	('does', 291)	('can', 49.21418012871439)	('ve', 18.270863639958304)	('can', 28.82297538311695)	('edu', 11.906194792288247)
17	('if', 1154)	('time', 282)	('with', 48.18761054093589)	('government', 18.089022419581	('are', 27.38799920531715)	('ve', 11.870735103204861)
18	('can', 1101)	('used', 272)	('are', 45.11043625995436)	('good', 17.967957804096248)	('with', 27.03257285835745)	('time', 11.745018935759806)
19	('as', 1026)	('ftp', 271)	('not', 43.53696710250099)	('bit', 17.440629791897027)	('not', 26.9403506133788)	('key', 11.670618169470233)

Рисунок 15 – Таблица результата векторизации для тестового набора данных без применения стемминга

	Count		TF		TF-IDF	
	Без стоп-слов	С стоп-словами	Без стоп-слов	С стоп-словами	Без стоп-слов	С стоп-словами
0	('the', 16688)	('thi', 2585)	('the', 561.4880559369901)	('thi', 169.76884542624322)	('the', 214.47110783764484)	('the', 214.47110783764484)
1	('to', 8883)	('use', 2014)	('to', 316.3652287700625)	('use', 116.9587739384261)	('to', 128.91799924792693)	('to', 128.91799924792693)
2	('of', 7021)	('key', 1283)	('of', 237.41198848231448)	('ani', 75.52328947198923)	('of', 101.9397995673158)	('of', 101.9397995673158)
3	('and', 6843)	('ha', 887)	('and', 212.34202585341927)	('key', 61.252598858687456)	('and', 91.85258731878157)	('and', 91.85258731878157)
4	('is', 5549)	('ani', 866)	('is', 185.57446689326932)	('wa', 61.24004770417701)	('is', 84.95276457618122)	('is', 84.95276457618122)
5	('in', 4419)	('wa', 783)	('it', 175.55160439683934)	('know', 58.981341558737796)	('it', 82.95304153471035)	('it', 82.95304153471035)
6	('it', 4191)	('encrypt', 774)	('in', 159.25693721881674)	('doe', 57.240041179177425)	('in', 73.08110129504044)	('in', 73.08110129504044)
7	('that', 3692)	('imag', 737)	('for', 145.53831889691298)	('like', 57.137968393386735)	('that', 73.01505717147099)	('that', 73.01505717147099)
8	('for', 3677)	('file', 730)	('that', 144.01545732916136)	('ha', 56.59160730937827)	('for', 68.08421571530532)	('for', 68.08421571530532)
9	('be', 2998)	('like', 711)	('you', 117.57659979606518)	('chip', 44.84136837996277)	('you', 66.66827307818957)	('you', 66.66827307818957)
10	('you', 2852)	('chip', 672)	('be', 104.11856816919685)	('just', 44.83029998661235)	('be', 59.109398696493265)	('be', 59.109398696493265)
11	('thi', 2585)	('doe', 671)	('thi', 103.06110204410142)	('thank', 41.73801066350643)	('thi', 55.45567314375222)	('thi', 55.45567314375222)
12	('on', 2459)	('know', 622)	('on', 94.5230013722625)	('work', 41.61901207046525)	('on', 50.853304634787605)	('on', 50.853304634787605)
13	('are', 2195)	('bit', 621)	('have', 89.88433983404117)	('anyon', 41.18249764862851)	('have', 50.30994107364992)	('have', 50.30994107364992)
14	('with', 2111)	('program', 569)	('with', 75.18149596453108)	('look', 40.281971512326024)	('are', 45.04857754230604)	('are', 45.04857754230604)
15	('or', 2090)	('db', 562)	('are', 74.93802609781713)	('file', 38.83230683543829)	('with', 44.02554259499939)	('with', 44.02554259499939)
16	('use', 2014)	('onli', 560)	('if', 74.24248878944846)	('need', 38.34165975253376)	('if', 43.91731908930183)	('if', 43.91731908930183)
17	('have', 1997)	('edu', 553)	('or', 74.23062669897583)	('encrypt', 36.15778681744015)	('or', 43.60061441479198)	('or', 43.60061441479198)
18	('as', 1784)	('data', 548)	('do', 71.9015042136159)	('onli', 34.91770247819603)	('do', 43.39462930056318)	('do', 43.39462930056318)
19	('not', 1740)	('secur', 534)	('use', 71.4761519919797)	('program', 33.24366815109684)	('use', 42.979583779815044)	('use', 42.979583779815044)

Рисунок 16 – Таблица результата векторизации для обучающего набора данных с применением стемминга

	Count		TF		TF-IDF	
	Без стоп-слов	С стоп-словами	Без стоп-слов	С стоп-словами	Без стоп-слов	С стоп-словами
0	('the', 9063)	('thi', 1472)	('the', 344.5586615001451)	('thi', 97.49594840488756)	('the', 133.4538752492904)	('thi', 97.49594840488756)
1	('to', 5360)	('use', 1097)	('to', 211.00830323462205)	('use', 71.55533662918268)	('to', 85.99632390188842)	('use', 71.55533662918268)
2	('of', 4137)	('imag', 998)	('of', 150.4948643044924)	('ani', 44.25521347634618)	('of', 65.18554426844001)	('ani', 44.25521347634618)
3	('and', 4073)	('file', 615)	('and', 137.86770734884337)	('know', 43.06179116896538)	('and', 59.63534071325143)	('know', 43.06179116896538)
4	('is', 3139)	('jpeg', 531)	('is', 121.90454220930677)	('wa', 39.53046546637826)	('it', 55.94816487314925)	('wa', 39.53046546637826)
5	('in', 2612)	('wa', 510)	('it', 114.00279976905662)	('like', 36.7672996472524)	('is', 55.94487331494873)	('like', 36.7672996472524)
6	('it', 2562)	('ani', 505)	('in', 102.13673946564619)	('ha', 34.4237897180173)	('that', 48.31727363290116)	('ha', 34.4237897180173)
7	('for', 2362)	('program', 497)	('that', 96.8675995748726)	('doe', 34.19333412527219)	('in', 47.30170083345097)	('doe', 34.19333412527219)
8	('that', 2237)	('ha', 479)	('for', 90.52244749715135)	('just', 29.513176624612083)	('you', 45.68450447600251)	('just', 29.513176624612083)
9	('you', 2086)	('edu', 468)	('you', 79.67798552554372)	('thank', 28.549402258522097)	('for', 43.47570804261264)	('thank', 28.549402258522097)
10	('be', 1647)	('like', 457)	('be', 68.85460938269497)	('anyon', 28.298548977191338)	('be', 37.344622742852174)	('anyon', 28.298548977191338)
11	('thi', 1472)	('bit', 451)	('on', 61.11286152167876)	('work', 26.494787158165767)	('thi', 34.3491285954964)	('work', 26.494787158165767)
12	('on', 1469)	('format', 411)	('have', 60.90208541002496)	('think', 24.640467411932118)	('have', 34.04975283966364)	('think', 24.640467411932118)
13	('have', 1298)	('know', 401)	('thi', 60.34198459030236)	('need', 24.56412549445321)	('on', 33.209071780256735)	('need', 24.56412549445321)
14	('or', 1295)	('doe', 386)	('or', 50.6519636067793)	('program', 24.56261574830257)	('if', 29.438597446898267)	('program', 24.56261574830257)
15	('with', 1260)	('data', 369)	('if', 49.388084620385364)	('look', 24.441912389427163)	('or', 29.11817727731375)	('look', 24.441912389427163)
16	('are', 1212)	('onli', 344)	('with', 47.226534170832274)	('make', 24.08146082002033)	('can', 28.41866932660334)	('make', 24.08146082002033)
17	('if', 1154)	('work', 344)	('can', 46.08463643662672)	('key', 23.553337023165067)	('are', 28.194428691654128)	('key', 23.553337023165067)
18	('use', 1097)	('make', 341)	('are', 45.258192235058026)	('pleas', 23.374159220189213)	('do', 27.903939642519983)	('pleas', 23.374159220189213)
19	('not', 1077)	('just', 339)	('do', 44.57070053438678)	('onli', 22.125484501080784)	('not', 27.900212314318484)	('onli', 22.125484501080784)

Рисунок 17 – Таблица результата векторизации для тестового набора данных с применением стемминга

Используя конвейер (Pipeline) реализуем модель наивного байесовского классификатора и выявим на основе показателей качества (значение полноты, точности, f1-меры и аккуратности), какая предварительная обработка данных обеспечит наилучшие результаты классификации. Полученный результат оптимальных параметров поиска представлен на рисунке 18.

```python
from sklearn.metrics import classification_report
from sklearn.feature_extraction.text import CountVectorizer, TfidfTransformer
from sklearn.naive_bayes import MultinomialNB
from sklearn.model_selection import GridSearchCV
from sklearn.pipeline import Pipeline

pipeline = Pipeline([
    ('vect', CountVectorizer()),
    ('tfidf', TfidfTransformer()),
    ('clf', MultinomialNB()),
])

parameters = {
    'vect__max_features': (500, 1000, 2500, 5000, 10000, None),
    'vect__stop_words': ('english', None),
    'tfidf__use_idf': (True, False),
}

grid_search = GridSearchCV(pipeline, parameters, n_jobs=-1, verbose=1)

grid_search.fit(train_bunch.data, train_bunch.target)

print("Best score: %0.3f" % grid_search.best_score_)
print("Best parameters set:")
grid_search.best_params_
```

Executed at 2023.11.12 13:28:06 in 4s 779ms

```
Fitting 5 folds for each of 24 candidates, totalling 120 fits
Best score: 0.893
Best parameters set:

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

Рисунок 18 – Результат классификации после нахождения оптимальных параметров через конвейер

Вывод

В ходе выполнения данной лабораторной работы я приобрел навыки предварительной обработки текстовых данных. В практической части исследования были использованы различные методы подсчета слов, включая как использование стемминга, так и без него. Кроме того, был применен метод векторизации с использованием TfidfTransformer с разными способами взвешивания. С использованием конвейера и сетки решений были найдены оптимальные наборы параметров для классификации, метрика которых базируется на оценках качества.

В результате исследования, наиболее лучшим способом предварительной обработки данных является векторизация TfidfTransformer с использованием TF-IDF взвешиваний и количество информативных терминов
= 5000.

Приложение А

Исходный код программы

Исходный код программы доступен по ссылке: https://github.com/ruslooob/expert-systems. В папке notebooks находятся файлы в формате ipynb, которые можно запустить у себя локально на компьютере.