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 Коровайцева А.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 №8

Классы 5, 16, 20

(comp.sys.mac.hardware, 'soc.religion.christian', 'talk.religion.misc')

Ход работы

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

```
categories = ['comp.sys.mac.hardware', 'soc.religion.christian', '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)
```

Рисунок 1 – Код для загрузки данных

После успешной загрузки данных посмотрим на записи. Для этого выведем по одному значению из обучаемой и тестовой выборки, которые представлены на рисунке 2.

```
twenty_train_full.data[0]
```

```
'A SIMM is a small PCB with DRAM chips soldered on.\n\n--maarten'
```

```
twenty_test_full.data[0]
```

```
"\nI don't know either.  Truth be known, so little is known of angels\nto even guess.  All we really know is that angels ALWAYS
speak in\nthe nativ tongue of the person they're talking to, so perhaps they\ndon't have ANY language of their own.\n\n\nWell,
we are told to test the spirits.  While you could do this\nscripturally, to see if someones claims are backed by the bible,\nI
see nothing wrong with making sure that that guy Lazarus really\nwas dead and now he's alive.\n\n\nIt's a common fallacy you co
mmit.  The non-falsifiability trick.  How\ncan I prove it when not all the evidence may be seen?  Answer:  I\ncan't.  The falla
cy is in assuming that it is up to me to prove \nanything.  \n\nWhen I say it has never been proven, I'm talking about the ones
\nmaking the claims, not the skeptics, who are doing the proving.\n\nThe burden of proof rest with the claimant.  Unfortunatel
y, \n(pontification warning) our legal system seems to be headed in\nthe dangerous realm of making people prove their innocence
(end\npontification).\n\nBut truthfully, Corinthians was so poorly written (or maybe just\nso poorly translated into English) t
hat much remains unknown\nabout just what Paul really intended (despite claims of hard\nproof one way or another).  Some will s
ee his writings in\n1 cor 12-14 as saying don't do this don't do this and using\nsarcasm, metaphor, etc. while yet others take
what he says literally\nsarcasms and metaphors notwithstanding."
```

Рисунок 2 – Пример загруженных данных

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

```
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
```

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

```
stem_train[0]
```

```
' a simm is a small pcb with dram chip solder on . -- maarten'
```

```
stem_test[0]
```

```
" i do n't know either . truth be known , so littl is known of angel to even guess . all we realli know is that angel alway spe
ak in the nativ tongu of the person they 're talk to , so perhap they do n't have ani languag of their own . well , we are told
to test the spirit . while you could do thi scriptur , to see if someon claim are back by the bibl , i see noth wrong with make
sure that that guy lazaru realli wa dead and now he 's aliv . it 's a common fallaci you commit . the non-falsifi trick . how c
an i prove it when not all the evid may be seen ? answer : i ca n't . the fallaci is in assum that it is up to me to prove anyt
h . when i say it ha never been proven , i 'm talk about the one make the claim , not the skeptic , who are do the prove . the
burden of proof rest with the claimant . unfortun , ( pontif warn ) our legal system seem to be head in the danger realm of mak
e peopl prove their innoc ( end pontif ) . but truth , corinthian wa so poorli written ( or mayb just so poorli translat into e
nglish ) that much remain unknown about just what paul realli intend ( despit claim of hard proof one way or anoth ) . some wil
l see hi write in 1 cor 12-14 as say do n't do thi do n't do thi and use sarcasm , metaphor , etc . while yet other take what h
e say liter sarcasm and metaphor notwithstand ."
```

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.

```
import numpy as np
from sklearn.feature_extraction.text import CountVectorizer
```

```
vect_without_stop = CountVectorizer(max_features=10000)
```

```
train_data = vect_without_stop.fit_transform(twenty_train_full.data)
test_data = vect_without_stop.transform(twenty_test_full.data)
```

```
def sort_by_tf(input_str):
    return input_str[1]

def top_terms(vector, data, count):
    x = list(zip(vector.get_feature_names_out(), np.ravel(data.sum(axis=0))))
    x.sort(key=sort_by_tf, reverse=True)
    return x[:count]
```

Рисунок 4 – Код для векторизации обучающей и тестовой выборки простым подсчетом слов

```
top_terms_without_stop = [{term[0]: term[1]} for term in top_terms(vect_without_stop, train_data, 20)]
top_terms_without_stop

top_terms_without_stop_test = [{term[0]: term[1]} for term in top_terms(vect_without_stop, test_data, 20)]
top_terms_without_stop_test
```

```
[{'the': 12380},
 {'to': 6270},
 {'of': 6251},
 {'and': 4764},
 {'that': 3928},
 {'is': 3902},
 {'in': 3771},
 {'it': 2684},
 {'you': 2165},
 {'not': 1933},
 {'for': 1839},
 {'this': 1688},
 {'be': 1600},
 {'as': 1579},
 {'are': 1534},
 {'have': 1479},
 {'with': 1475},
 {'on': 1351},
 {'but': 1136},
 {'or': 1132}]
```

Рисунок 5 – Результат векторизации обучающей и 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_stop = CountVectorizer(max_features=10000, stop_words='english')
```

```
train_data_stop = vect_stop.fit_transform(twenty_train_full.data)
test_data_stop = vect_stop.transform(twenty_test_full.data)
```

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

```
top_terms_stop = [{term[0]: term[1]} for term in top_terms(vect_stop, train_data_stop, 20)]
top_terms_stop

top_terms_stop_test = [{term[0]: term[1]} for term in top_terms(vect_stop, test_data_stop, 20)]
top_terms_stop_test
```

```
[{'god': 1108},
 {'people': 471},
 {'know': 424},
 {'don': 385},
 {'just': 380},
 {'does': 378},
 {'like': 375},
 {'jesus': 368},
 {'christ': 367},
 {'think': 343},
 {'church': 307},
 {'time': 301},
 {'lord': 298},
 {'say': 287},
 {'did': 269},
 {'christian': 268},
 {'bible': 267},
 {'believe': 264},
 {'sin': 259},
 {'mac': 255}]
```

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_stem_without_stop = CountVectorizer(max_features=10000)
```

```
train_data_without_stop_stem = vect_stem_without_stop.fit_transform(stem_train)
test_data_without_stop_stem = vect_stem_without_stop.transform(stem_test)
```

```
top_terms_stem = [{term[0]: term[1]} for term in top_terms(vect_stem_without_stop, train_data_without_stop_stem, 20)]
top_terms_stem

top_terms_stem_test = [{term[0]: term[1]} for term in top_terms(vect_stem_without_stop, test_data_without_stop_stem, 20)]
top_terms_stem_test
```

```
[{'the': 12380},
 {'to': 6270},
 {'of': 6251},
 {'and': 4764},
 {'is': 3955},
 {'that': 3930},
 {'in': 3771},
 {'it': 2882},
 {'you': 2165},
 {'not': 2042},
 {'be': 1904},
 {'for': 1839},
 {'thi': 1756},
 {'have': 1614},
 {'as': 1579},
 {'are': 1558},
 {'with': 1476},
 {'on': 1354},
 {'do': 1217},
 {'god': 1173}]
```

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

```
vect_stem = CountVectorizer(max_features=10000, stop_words='english')
```

```
train_data_stop_stem = vect_stem.fit_transform(stem_train)
test_data_stop_stem = vect_stem.transform(stem_test)
```

```
top_terms_stop_stem = [{term[0]: term[1]} for term in top_terms(vect_stem, train_data_stop_stem, 20)]
top_terms_stop_stem

top_terms_stop_stem_test = [{term[0]: term[1]} for term in top_terms(vect_stem, test_data_stop_stem, 20)]
top_terms_stop_stem_test
```

```
[{'thi': 1756},
 {'god': 1173},
 {'wa': 1154},
 {'hi': 737},
 {'christian': 624},
 {'ha': 566},
 {'use': 563},
 {'ani': 510},
 {'doe': 510},
 {'say': 487},
 {'know': 485},
 {'peopl': 473},
 {'like': 425},
 {'homosexu': 410},
 {'sin': 396},
 {'onli': 390},
 {'just': 380},
 {'think': 375},
 {'believ': 371},
 {'christ': 368}]
```

Рисунок 9 – Результат векторизации обучающей и тестовой 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 стопслов представлен на рисунке 10, с использованием стоп-слов представлен на рисунке 11.

```
from sklearn.feature_extraction.text import TfidfTransformer
```

```
tf = TfidfTransformer(use_idf=False)
tfidf = TfidfTransformer(use_idf=True)
```

```
train_data_tf = tf.fit_transform(train_data)
test_data_tf = tf.transform(test_data)

train_data_tfidf = tfidf.fit_transform(train_data)
test_data_tfidf = tfidf.transform(test_data)
```

```
top_terms_tf = [{term[0]: term[1]} for term in top_terms(vect_without_stop, train_data_tf, 20)]
top_terms_tf

top_terms_tf_test = [{term[0]: term[1]} for term in top_terms(vect_without_stop, test_data_tf, 20)]
top_terms_tf_test

top_terms_tfidf = [{term[0]: term[1]} for term in top_terms(vect_without_stop, train_data_tfidf, 20)]
top_terms_tfidf

top_terms_tfidf_test = [{term[0]: term[1]} for term in top_terms(vect_without_stop, test_data_tfidf, 20)]
top_terms_tfidf_test
```

```
[{'the': 148.86207833224387},
 {'to': 86.24069806048266},
 {'of': 78.2528272131044},
 {'and': 62.57405232529181},
 {'that': 61.91582724132609},
 {'is': 58.04952814992369},
 {'in': 54.30601035917385},
 {'it': 50.695055313142554},
 {'you': 47.66881664363077},
 {'not': 35.309183144321366},
 {'for': 35.10046683041504},
 {'this': 32.70420270208634},
 {'have': 31.709762102532206},
 {'be': 30.59476801700976},
 {'on': 29.451538737736513},
 {'with': 29.234115766232215},
 {'are': 28.848411519142573},
 {'as': 28.23696021657093},
 {'was': 27.34951804295569},
 {'or': 25.671178548376496}]
```

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

```
tf = TfidfTransformer(use_idf=False)
tfidf = TfidfTransformer(use_idf=True)
```

```
train_data_stop_tf = tf.fit_transform(train_data_stop)
test_data_stop_tf = tf.transform(test_data_stop)

train_data_stop_tfidf = tfidf.fit_transform(train_data_stop)
test_data_stop_tfidf = tfidf.transform(test_data_stop)
```

```
top_terms_stop_tf = [{term[0]: term[1]} for term in top_terms(vect_stop, train_data_stop_tf, 20)]
top_terms_stop_tf

top_terms_stop_tf_test = [{term[0]: term[1]} for term in top_terms(vect_stop, test_data_stop_tf, 20)]
top_terms_stop_tf_test

top_terms_stop_tfidf = [{term[0]: term[1]} for term in top_terms(vect_stop, train_data_stop_tfidf, 20)]
top_terms_stop_tfidf

top_terms_stop_tfidf_test = [{term[0]: term[1]} for term in top_terms(vect_stop, test_data_stop_tfidf, 20)]
top_terms_stop_tfidf_test
```

```
[{'god': 27.95497268349866},
 {'know': 18.542141116243307},
 {'just': 16.210267240880423},
 {'does': 16.137672246629286},
 {'don': 16.033474930641432},
 {'like': 15.537254867563917},
 {'think': 14.97705548765351},
 {'people': 14.195382946062232},
 {'mac': 13.981210612738836},
 {'jesus': 12.817753218849594},
 {'christian': 12.506478769588576},
 {'church': 12.347646222961615},
 {'did': 11.951076435168872},
 {'sin': 11.922112737935514},
 {'apple': 11.865499882550811},
 {'monitor': 11.667768613399344},
 {'time': 11.52942804011032},
 {'read': 10.626657803538777},
 {'christ': 10.570073397684714},
 {'believe': 10.530471367142}]
```

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.

```
tf = TfidfTransformer(use_idf=False)
tfidf = TfidfTransformer(use_idf=True)
```

```
train_data_stem_tf = tf.fit_transform(train_data_without_stop_stem)
test_data_stem_tf = tf.transform(test_data_without_stop_stem)

train_data_stem_tfidf = tfidf.fit_transform(train_data_without_stop_stem)
test_data_stem_tfidf = tfidf.transform(test_data_without_stop_stem)
```

```
top_terms_stem_tf = [{term[0]: term[1]} for term in top_terms(vect_stem_without_stop, train_data_stem_tf, 20)]
top_terms_stem_tf

top_terms_stem_tf_test = [{term[0]: term[1]} for term in top_terms(vect_stem_without_stop, test_data_stem_tf, 20)]
top_terms_stem_tf_test

top_terms_stem_tfidf = [{term[0]: term[1]} for term in top_terms(vect_stem_without_stop, train_data_stem_tfidf, 20)]
top_terms_stem_tfidf

top_terms_stem_tfidf_test = [{term[0]: term[1]} for term in top_terms(vect_stem_without_stop, test_data_stem_tfidf, 20)]
top_terms_stem_tfidf_test
```

```
[{'the': 148.44840163307802},
 {'to': 86.18185750706422},
 {'of': 77.96764025564627},
 {'and': 62.34718973797119},
 {'that': 61.801008270307904},
 {'is': 58.55502667880058},
 {'in': 54.22442872955803},
 {'it': 52.874155787450924},
 {'you': 47.98992721956406},
 {'not': 36.69440941285236},
 {'for': 35.22027869058943},
 {'be': 34.71864300028177},
 {'have': 34.241537522424174},
 {'thi': 33.153367265959325},
 {'on': 29.48488130877514},
 {'with': 29.160656939048575},
 {'are': 28.988578800351764},
 {'do': 28.875533246770882},
 {'as': 27.99019444534268},
 {'wa': 27.759404376818473}]
```

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

```
tf = TfidfTransformer(use_idf=False)
tfidf = TfidfTransformer(use_idf=True)
```

```
train_data_stem_stop_tf = tf.fit_transform(train_data_stop_stem)
test_data_stem_stop_tf = tf.transform(test_data_stop_stem)

train_data_stem_stop_tfidf = tfidf.fit_transform(train_data_stop_stem)
test_data_stem_stop_tfidf = tfidf.transform(test_data_stop_stem)
```

```
top_terms_stem_stop_tf = [{term[0]: term[1]} for term in top_terms(vect_stem, train_data_stop_tf, 20)]
top_terms_stem_stop_tf

top_terms_stem_stop_tf_test = [{term[0]: term[1]} for term in top_terms(vect_stem, test_data_stop_tf, 20)]
top_terms_stem_stop_tf_test

top_terms_stem_stop_tfidf = [{term[0]: term[1]} for term in top_terms(vect_stem, train_data_stop_tf, 20)]
top_terms_stem_stop_tfidf

top_terms_stem_stop_tfidf_test = [{term[0]: term[1]} for term in top_terms(vect_stem, test_data_stop_tf, 20)]
top_terms_stem_stop_tfidf_test
```

```
[{'hazrat': 52.56617814468827},
 {'merchandis': 37.79665379581945},
 {'mask': 32.38004562022943},
 {'dieti': 31.714807369343013},
 {'di': 31.552249334010018},
 {'muham': 29.99492088865641},
 {'undiscuss': 27.996778310542236},
 {'relativli': 27.570418756826566},
 {'nobodi': 21.00989194778251},
 {'uniti': 20.69403326971876},
 {'magisterieum': 20.010418361950915},
 {'automat': 19.476981780463902},
 {'cyru': 18.95344612955057},
 {'stare': 18.273135374257958},
 {'aviat': 17.889577615820613},
 {'wish': 17.380657714430487},
 {'abomin': 17.14902573648428},
 {'795': 16.935555097962848},
 {'prone': 16.60129290675252},
 {'sgi': 16.25414677898279}]
```

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': 16652}	{'god': 1427}	{'the': 543.0887145775569}	{'god': 84.04605919516031}	{'the': 215.18803930002346}	{'god': 43.82121116891297}
1	{'to': 8490}	{'people': 779}	{'to': 286.0938629517631}	{'know': 53.06617041632788}	{'to': 122.86230443612654}	{'jesus': 26.401690904264303}
2	{'of': 8334}	{'jesus': 722}	{'of': 249.31964725391921}	{'people': 51.461500350700305}	{'of': 113.2408746455194}	{'people': 26.116384920488024}
3	{'and': 6656}	{'know': 625}	{'and': 209.99355122874857}	{'just': 49.95803872415939}	{'and': 96.00866795656545}	{'know': 25.990836762272973}
4	{'that': 5747}	{'does': 624}	{'is': 194.50478082296198}	{'does': 47.66231977464952}	{'that': 90.94335853814704}	{'just': 24.554027466416436}
5	{'is': 5591}	{'just': 586}	{'that': 185.161466645549}	{'don': 45.904856306460154}	{'is': 89.68245849269316}	{'does': 24.159380757560697}
6	{'in': 4801}	{'don': 571}	{'in': 155.5536983679511}	{'like': 44.557247980439584}	{'in': 75.20695624577941}	{'don': 23.236656368031195}
7	{'it': 3830}	{'think': 565}	{'it': 141.18074652398246}	{'think': 42.683000298870574}	{'it': 70.64716108495936}	{'like': 22.53936626597992}
8	{'you': 3092}	{'like': 559}	{'you': 117.98544884456389}	{'jesus': 41.40465459972886}	{'you': 70.267094812125}	{'think': 22.101945828068693}
9	{'not': 2749}	{'say': 461}	{'for': 105.73857309057988}	{'believe': 31.590391737280633}	{'for': 53.56843356240359}	{'mac': 19.43308741744436}
10	{'for': 2699}	{'time': 444}	{'this': 89.73752243825957}	{'time': 31.41388463517261}	{'this': 50.75562852683706}	{'believe': 19.378790437446035}
11	{'this': 2486}	{'believe': 437}	{'not': 86.13716438441566}	{'good': 29.98678066435632}	{'not': 50.09016544368851}	{'christian': 17.988599957072733}
12	{'be': 2316}	{'good': 416}	{'be': 84.021549942572}	{'say': 29.883715281817828}	{'be': 47.79328531321388}	{'say': 17.324507131690708}
13	{'are': 2220}	{'church': 414}	{'have': 82.5414991960181}	{'mac': 29.593860305546464}	{'are': 45.58237270450893}	{'good': 17.28507445229634}
14	{'have': 2166}	{'bible': 411}	{'with': 78.72180061552618}	{'christian': 28.097482333068694}	{'have': 45.302196085167424}	{'time': 17.23652678997511}
15	{'as': 2136}	{'christian': 396}	{'are': 75.29563703121633}	{'use': 27.373307702686343}	{'with': 42.98742534542351}	{'christians': 17.137530582992603}
16	{'with': 2071}	{'way': 377}	{'on': 70.41570789025904}	{'way': 25.76215334315875}	{'as': 41.569249901325044}	{'bible': 16.90297950194337}
17	{'on': 1823}	{'christ': 373}	{'if': 65.83106887666393}	{'problem': 25.573089707965188}	{'on': 39.87771437760014}	{'apple': 16.555927890043925}
18	{'but': 1818}	{'did': 373}	{'but': 64.7675841529437}	{'christians': 25.446644491594235}	{'if': 38.47381716307315}	{'church': 16.50243190133636}
19	{'was': 1622}	{'christians': 333}	{'as': 63.60347816394421}	{'bible': 25.228516436347924}	{'but': 38.129554444839656}	{'thanks': 16.34820801151869}

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': 12380}	{'god': 1108}	{'the': 364.8095702637773}	{'god': 52.56617814468827}	{'the': 148.86207833224387}	{'god': 27.95497268349866}
1	{'to': 6270}	{'people': 471}	{'to': 197.36462753116095}	{'know': 37.79665379581945}	{'to': 86.24069806048266}	{'know': 18.542141116243307}
2	{'of': 6251}	{'know': 424}	{'of': 166.55696621672018}	{'just': 32.38004562022943}	{'of': 78.2528272131044}	{'just': 16.210267240880423}
3	{'and': 4764}	{'don': 385}	{'and': 133.468347925069}	{'don': 31.714807369343013}	{'and': 62.57405232529181}	{'does': 16.137672246629286}
4	{'that': 3928}	{'just': 380}	{'that': 124.37884585010595}	{'does': 31.552249334010018}	{'that': 61.91582724132609}	{'don': 16.033474930641432}
5	{'is': 3902}	{'does': 378}	{'is': 121.71255438787193}	{'like': 29.99492088865641}	{'is': 58.04952814992369}	{'like': 15.537254867563917}
6	{'in': 3771}	{'like': 375}	{'in': 107.7277527155332}	{'think': 27.996778310542236}	{'in': 54.30601035917385}	{'think': 14.97705548765351}
7	{'it': 2684}	{'jesus': 368}	{'it': 99.81441760032114}	{'people': 27.570418756826566}	{'it': 50.695055313142554}	{'people': 14.195382946062232}
8	{'you': 2165}	{'christ': 367}	{'you': 79.20779132071978}	{'mac': 21.00989194778251}	{'you': 47.66881664363077}	{'mac': 13.981210612738836}
9	{'not': 1933}	{'think': 343}	{'for': 67.78077562867219}	{'time': 20.694033226971876}	{'not': 35.309183144321366}	{'jesus': 12.817753218849594}
10	{'for': 1839}	{'church': 307}	{'not': 59.84862781060529}	{'jesus': 20.010418361950915}	{'for': 35.10046683041504}	{'christian': 12.506478769588576}
11	{'this': 1688}	{'time': 301}	{'have': 56.67244054149067}	{'christian': 19.476981780463902}	{'this': 32.70420270208634}	{'church': 12.347646222961615}
12	{'be': 1600}	{'lord': 298}	{'this': 56.440703899829245}	{'did': 18.95344612955057}	{'have': 31.709762102532206}	{'did': 11.951076435168872}
13	{'as': 1579}	{'say': 287}	{'be': 52.411421997778284}	{'say': 18.273135374257958}	{'be': 30.59476801700976}	{'sin': 11.922112737935514}
14	{'are': 1534}	{'did': 269}	{'on': 51.36403034174104}	{'church': 17.889577615820613}	{'on': 29.451538737736513}	{'apple': 11.865499882550811}
15	{'have': 1479}	{'christian': 268}	{'with': 50.75947933208338}	{'way': 17.380657714430487}	{'with': 29.234115766232215}	{'monitor': 11.667768613399344}
16	{'with': 1475}	{'bible': 267}	{'are': 45.60043873793938}	{'believe': 17.14902573648428}	{'are': 28.848411519142573}	{'time': 11.52942804011032}
17	{'on': 1351}	{'believe': 264}	{'as': 41.77266737924603}	{'apple': 16.935555097962848}	{'as': 28.23696021657093}	{'read': 10.626657803538777}
18	{'but': 1136}	{'sin': 259}	{'if': 39.598719691654146}	{'new': 16.60129290675252}	{'was': 27.34951804295569}	{'christ': 10.570073397684714}
19	{'or': 1132}	{'mac': 255}	{'but': 39.57916126789582}	{'read': 16.25414677898279}	{'or': 25.671178548376496}	{'believe': 10.530471367142}

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': 16651}	{'thi': 2494}	{'the': 533.6024042368656}	{'hazrat': 84.04605919516031}	{'the': 216.64166617238433}	{'hazrat': 84.04605919516031}
1	{'to': 8490}	{'wa': 1669}	{'to': 280.79790354201583}	{'merchandis': 53.06617041632788}	{'to': 124.00623302448895}	{'merchandis': 53.06617041632788}
2	{'of': 8334}	{'god': 1453}	{'of': 244.9348757517005}	{'relativli': 51.461500350700305}	{'of': 114.05132687333837}	{'relativli': 51.461500350700305}
3	{'and': 6657}	{'hi': 1004}	{'and': 206.1233106003709}	{'mask': 49.95803872415939}	{'and': 96.67208703969094}	{'mask': 49.95803872415939}
4	{'that': 5748}	{'christian': 909}	{'is': 194.3722816360924}	{'di': 47.66231977464952}	{'that': 91.9081519574775}	{'di': 47.66231977464952}
5	{'is': 5686}	{'ha': 867}	{'that': 181.67024370851857}	{'dieti': 45.904856306460154}	{'is': 91.65225243008739}	{'dieti': 45.904856306460154}
6	{'in': 4801}	{'doe': 788}	{'in': 152.53476815361205}	{'muham': 44.557247980439584}	{'in': 75.5050120617292}	{'muham': 44.557247980439584}
7	{'it': 4087}	{'peopl': 785}	{'it': 146.0633678113764}	{'undiscuss': 42.683000298870574}	{'it': 74.18016258094475}	{'undiscuss': 42.683000298870574}
8	{'you': 3091}	{'say': 759}	{'you': 115.83370128734639}	{'magisterieum': 41.40465459972886}	{'you': 70.98559082363474}	{'magisterieum': 41.40465459972886}
9	{'not': 2921}	{'use': 731}	{'for': 103.80770583846451}	{'abomin': 31.590391737280633}	{'for': 54.01225755326821}	{'abomin': 31.590391737280633}
10	{'be': 2723}	{'know': 720}	{'be': 93.93792886919096}	{'uniti': 31.41388463517261}	{'be': 53.347885440854434}	{'uniti': 31.41388463517261}
11	{'for': 2699}	{'jesu': 716}	{'not': 89.13474917240082}	{'heat': 29.98678066435632}	{'not': 52.9299304720662}	{'heat': 29.98678066435632}
12	{'thi': 2494}	{'think': 659}	{'thi': 88.44240833356106}	{'stare': 29.883715281817828}	{'thi': 51.466855199772894}	{'stare': 29.883715281817828}
13	{'have': 2332}	{'ani': 658}	{'have': 87.30612987422288}	{'nobodi': 29.593860305546464}	{'have': 48.22745458111488}	{'nobodi': 29.593860305546464}
14	{'are': 2261}	{'onli': 625}	{'with': 77.19469611884804}	{'automat': 28.097482333068694}	{'are': 46.82287154841494}	{'automat': 28.097482333068694}
15	{'as': 2134}	{'like': 613}	{'are': 75.68404303725856}	{'vp': 27.373307702686343}	{'with': 43.31005469740532}	{'vp': 27.373307702686343}
16	{'with': 2071}	{'believ': 599}	{'on': 69.420258136969494}	{'wish': 25.762153343315875}	{'as': 41.801909793427605}	{'wish': 25.762153343315875}
17	{'do': 1828}	{'just': 586}	{'do': 65.4990726142797}	{'sall': 25.573089707965188}	{'do': 41.33281243586461}	{'sall': 25.573089707965188}
18	{'on': 1828}	{'time': 532}	{'if': 64.51634873919015}	{'avail': 25.446644491594235}	{'on': 40.45808721538086}	{'avail': 25.446644491594235}
19	{'but': 1818}	{'did': 518}	{'but': 63.585352804666755}	{'acceptab': 25.228516436347924}	{'if': 38.77263584827218}	{'acceptab': 25.228516436347924}

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': 12380}	{'thi': 1756}	{'the': 357.42953269535775}	{'hazrat': 52.56617814468827}	{'the': 148.44840163307802}	{'hazrat': 52.56617814468827}
1	{'to': 6270}	{'god': 1173}	{'to': 193.0934678602922}	{'merchandis': 37.79665379581945}	{'to': 86.18185750706422}	{'merchandis': 37.79665379581945}
2	{'of': 6251}	{'wa': 1154}	{'of': 163.11159230069902}	{'mask': 32.38004562022943}	{'of': 77.96764025564627}	{'mask': 32.38004562022943}
3	{'and': 4764}	{'hi': 737}	{'and': 130.64064754373393}	{'dieti': 31.714807369343013}	{'and': 62.34718973797119}	{'dieti': 31.714807369343013}
4	{'is': 3955}	{'christian': 624}	{'that': 121.85582499990868}	{'di': 31.552249334010018}	{'that': 61.801008270307904}	{'di': 31.552249334010018}
5	{'that': 3930}	{'ha': 566}	{'is': 121.07210432311149}	{'muham': 29.99492088865641}	{'is': 58.55502667880058}	{'muham': 29.99492088865641}
6	{'in': 3771}	{'use': 563}	{'in': 105.56690438668741}	{'undiscuss': 27.996778310542236}	{'in': 54.22442872955803}	{'undiscuss': 27.996778310542236}
7	{'it': 2882}	{'ani': 510}	{'it': 102.97217168052809}	{'relativli': 27.570418756826566}	{'it': 52.874155787450924}	{'relativli': 27.570418756826566}
8	{'you': 2165}	{'doe': 510}	{'you': 77.64234325340601}	{'nobodi': 21.00989194778251}	{'you': 47.98992721956406}	{'nobodi': 21.00989194778251}
9	{'not': 2042}	{'say': 487}	{'for': 66.44039176340249}	{'uniti': 20.69403326971876}	{'not': 36.69440941285236}	{'uniti': 20.69403326971876}
10	{'be': 1904}	{'know': 485}	{'not': 61.51670078442213}	{'magisterieum': 20.010418361950915}	{'for': 35.22027869058943}	{'magisterieum': 20.010418361950915}
11	{'for': 1839}	{'peopl': 473}	{'have': 61.345637245795935}	{'automat': 19.476981780463902}	{'be': 34.71864300028177}	{'automat': 19.476981780463902}
12	{'thi': 1756}	{'like': 425}	{'be': 60.1583338096674}	{'cyru': 18.95344612955057}	{'have': 34.241537522424174}	{'cyru': 18.95344612955057}
13	{'have': 1614}	{'homosexu': 410}	{'thi': 55.97694335313927}	{'stare': 18.273135374257958}	{'thi': 33.153367265959325}	{'stare': 18.273135374257958}
14	{'as': 1579}	{'sin': 396}	{'on': 50.23064353616814}	{'aviat': 17.889577615820613}	{'on': 29.48488130877514}	{'aviat': 17.889577615820613}
15	{'are': 1558}	{'onli': 390}	{'with': 49.68487967865335}	{'wish': 17.380657714430487}	{'with': 29.160656939048575}	{'wish': 17.380657714430487}
16	{'with': 1476}	{'just': 380}	{'are': 45.2643257420303}	{'abomin': 17.14902573648428}	{'are': 28.988578800351764}	{'abomin': 17.14902573648428}
17	{'on': 1354}	{'think': 375}	{'do': 44.96495617079795}	{'795': 16.935555097962848}	{'do': 28.875533246770882}	{'795': 16.935555097962848}
18	{'do': 1217}	{'believ': 371}	{'as': 40.80538032516044}	{'prone': 16.60129290675252}	{'as': 27.99019444534268}	{'prone': 16.60129290675252}
19	{'god': 1173}	{'christ': 368}	{'if': 38.80762817439416}	{'sgi': 16.25414677898279}	{'wa': 27.759404376818473}	{'sgi': 16.25414677898279}

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

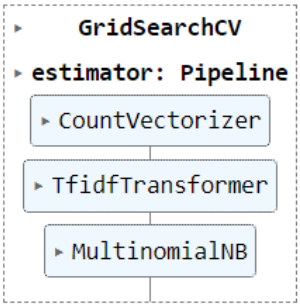
Используя конвейер (Pipeline) реализуем модель наивного байесовского классификатора и выявим на основе показателей качества (значение полноты, точности, f1-меры и аккуратности), какая предварительная обработка данных обеспечит наилучшие результаты классификации. Полученный результат оптимальных параметров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

```
from sklearn.metrics import classification_report
from sklearn.naive_bayes import MultinomialNB
```

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

```
from sklearn.model_selection import GridSearchCV

gscv = GridSearchCV(text_clf, param_grid=parameters)
gscv.fit(twenty_train_full.data, twenty_train_full.target)
```




```
print(classification_report(gscv.predict(twenty_test_full.data), twenty_test_full.target))
```

```
              precision    recall  f1-score   support

           0       0.94      0.94      0.94       386
           1       0.95      0.65      0.77       583
           2       0.23      0.88      0.36        65

    accuracy                           0.77      1034
   macro avg       0.71      0.82      0.69      1034
weighted avg       0.90      0.77      0.81      1034
```

```
gscv.best_params_
```

```
{'tfidf__use_idf': True,
 'vect__max_features': 2000,
 'vect__stop_words': 'english'}
```

Рисунок 18 – Результат классификации после нахождения оптимальных параметров через конвейер

Вывод

В ходе выполнения данной лабораторной работы мною были получены навыки предварительной обработки текстовых данных. В практической части исследования были использованы различные методы подсчета слов, включая как использование стемминга, так и без него.

Также 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 было выявлено, что наиболее лучшим способом предварительной обработки данных является векторизация TfidfTransformer с использованием TF-IDF взвешиваний и количество информативных терминов = 2000.

Приложение А

Исходный код

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

# # Лабораторная работа №2
# ### Задание
# ### Вариант №8
# ### Классы 5, 16, 20 ('comp.sys.mac.hardware', ''soc.religion.christian',
''talk.religion.misc')

# In[1]:


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


# In[2]:


categories = ['comp.sys.mac.hardware', 'soc.religion.christian',
'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)


# In[3]:


twenty_train_full.data[0]


# In[4]:


twenty_test_full.data[0]


# ### Применение стемминга

# In[54]:


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

nltk.download('punkt')


# In[6]:


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[7]:


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


# In[8]:


stem_train[0]


# In[9]:


stem_test[0]


# ### Векторизация выборки

# #### Векторизация обучающей и тестовой выборки простым подсчетом слов
(CountVectorizer) и значением max_features = 10.000

# In[10]:


import numpy as np
from sklearn.feature_extraction.text import CountVectorizer


# In[11]:


vect_without_stop = CountVectorizer(max_features=10000)


# In[12]:


train_data = vect_without_stop.fit_transform(twenty_train_full.data)
test_data = vect_without_stop.transform(twenty_test_full.data)


# In[13]:


def sort_by_tf(input_str):
    return input_str[1]

def top_terms(vector, data, count):
    x = list(zip(vector.get_feature_names_out(), np.ravel(data.sum(axis=0))))
    x.sort(key=sort_by_tf, reverse=True)
    return x[:count]
```

```
# In[14]:


top_terms_without_stop = [{term[0]: term[1]} for term in
top_terms(vect_without_stop, train_data, 20)]
top_terms_without_stop

top_terms_without_stop_test = [{term[0]: term[1]} for term in
top_terms(vect_without_stop, test_data, 20)]
top_terms_without_stop_test


# ### Отсечение стоп-слов

# In[15]:


vect_stop = CountVectorizer(max_features=10000, stop_words='english')


# In[16]:


train_data_stop = vect_stop.fit_transform(twenty_train_full.data)
test_data_stop = vect_stop.transform(twenty_test_full.data)


# In[17]:


top_terms_stop = [{term[0]: term[1]} for term in top_terms(vect_stop,
train_data_stop, 20)]
top_terms_stop

top_terms_stop_test = [{term[0]: term[1]} for term in top_terms(vect_stop,
test_data_stop, 20)]
top_terms_stop_test


# ### Для данных после стемминга

# #### Без стоп-слов

# In[18]:


vect_stem_without_stop = CountVectorizer(max_features=10000)


# In[19]:


train_data_without_stop_stem =
vect_stem_without_stop.fit_transform(stem_train)
test_data_without_stop_stem = vect_stem_without_stop.transform(stem_test)


# In[20]:


top_terms_stem = [{term[0]: term[1]} for term in
top_terms(vect_stem_without_stop, train_data_without_stop_stem, 20)]
top_terms_stem
```

```
top_terms_stem_test = [{term[0]: term[1]} for term in
top_terms(vect_stem_without_stop, test_data_without_stop_stem, 20)]
top_terms_stem_test


# #### С использованием стоп-слов

# In[21]:


vect_stem = CountVectorizer(max_features=10000, stop_words='english')


# In[22]:


train_data_stop_stem = vect_stem.fit_transform(stem_train)
test_data_stop_stem = vect_stem.transform(stem_test)


# In[23]:


top_terms_stop_stem = [{term[0]: term[1]} for term in top_terms(vect_stem,
train_data_stop_stem, 20)]
top_terms_stop_stem

top_terms_stop_stem_test = [{term[0]: term[1]} for term in
top_terms(vect_stem, test_data_stop_stem, 20)]
top_terms_stop_stem_test


# ### Векторизация выборки с помощью TfidfTransformer (TF и TF-IDF)

# #### Без использования стоп-слов

# In[24]:


from sklearn.feature_extraction.text import TfidfTransformer


# In[25]:


tf = TfidfTransformer(use_idf=False)
tfidf = TfidfTransformer(use_idf=True)


# In[26]:


train_data_tf = tf.fit_transform(train_data)
test_data_tf = tf.transform(test_data)

train_data_tfidf = tfidf.fit_transform(train_data)
test_data_tfidf = tfidf.transform(test_data)


# In[27]:


top_terms_tf = [{term[0]: term[1]} for term in top_terms(vect_without_stop,
```

```
train_data_tf, 20)]
top_terms_tf

top_terms_tf_test = [{term[0]: term[1]} for term in
top_terms(vect_without_stop, test_data_tf, 20)]
top_terms_tf_test

top_terms_tfidf = [{term[0]: term[1]} for term in
top_terms(vect_without_stop, train_data_tfidf, 20)]
top_terms_tfidf

top_terms_tfidf_test = [{term[0]: term[1]} for term in
top_terms(vect_without_stop, test_data_tfidf, 20)]
top_terms_tfidf_test


# #### С использованием стоп-слов

# In[28]:


tf = TfidfTransformer(use_idf=False)
tfidf = TfidfTransformer(use_idf=True)


# In[29]:


train_data_stop_tf = tf.fit_transform(train_data_stop)
test_data_stop_tf = tf.transform(test_data_stop)

train_data_stop_tfidf = tfidf.fit_transform(train_data_stop)
test_data_stop_tfidf = tfidf.transform(test_data_stop)


# In[30]:


top_terms_stop_tf = [{term[0]: term[1]} for term in top_terms(vect_stop,
train_data_stop_tf, 20)]
top_terms_stop_tf

top_terms_stop_tf_test = [{term[0]: term[1]} for term in top_terms(vect_stop,
test_data_stop_tf, 20)]
top_terms_stop_tf_test

top_terms_stop_tfidf = [{term[0]: term[1]} for term in top_terms(vect_stop,
train_data_stop_tfidf, 20)]
top_terms_stop_tfidf

top_terms_stop_tfidf_test = [{term[0]: term[1]} for term in
top_terms(vect_stop, test_data_stop_tfidf, 20)]
top_terms_stop_tfidf_test


# #### Со стеммингом без стоп-слов

# In[31]:


tf = TfidfTransformer(use_idf=False)
tfidf = TfidfTransformer(use_idf=True)
```

```
# In[32]:


train_data_stem_tf = tf.fit_transform(train_data_without_stop_stem)
test_data_stem_tf = tf.transform(test_data_without_stop_stem)

train_data_stem_tfidf = tfidf.fit_transform(train_data_without_stop_stem)
test_data_stem_tfidf = tfidf.transform(test_data_without_stop_stem)


# In[33]:


top_terms_stem_tf = [{term[0]: term[1]} for term in
top_terms(vect_stem_without_stop, train_data_stem_tf, 20)]
top_terms_stem_tf

top_terms_stem_tf_test = [{term[0]: term[1]} for term in
top_terms(vect_stem_without_stop, test_data_stem_tf, 20)]
top_terms_stem_tf_test

top_terms_stem_tfidf = [{term[0]: term[1]} for term in
top_terms(vect_stem_without_stop, train_data_stem_tfidf, 20)]
top_terms_stem_tfidf

top_terms_stem_tfidf_test = [{term[0]: term[1]} for term in
top_terms(vect_stem_without_stop, test_data_stem_tfidf, 20)]
top_terms_stem_tfidf_test


# #### Со стеммингом с использованием стоп-слов

# In[34]:


tf = TfidfTransformer(use_idf=False)
tfidf = TfidfTransformer(use_idf=True)


# In[35]:


train_data_stem_stop_tf = tf.fit_transform(train_data_stop_stem)
test_data_stem_stop_tf = tf.transform(test_data_stop_stem)

train_data_stem_stop_tfidf = tfidf.fit_transform(train_data_stop_stem)
test_data_stem_stop_tfidf = tfidf.transform(test_data_stop_stem)


# In[36]:


top_terms_stem_stop_tf = [{term[0]: term[1]} for term in top_terms(vect_stem,
train_data_stop_tf, 20)]
top_terms_stem_stop_tf

top_terms_stem_stop_tf_test = [{term[0]: term[1]} for term in
top_terms(vect_stem, test_data_stop_tf, 20)]
top_terms_stem_stop_tf_test

top_terms_stem_stop_tfidf = [{term[0]: term[1]} for term in
top_terms(vect_stem, train_data_stop_tf, 20)]
top_terms_stem_stop_tfidf
```

```
top_terms_stem_stop_tfidf_test = [{term[0]: term[1]} for term in
top_terms(vect_stem, test_data_stop_tf, 20)]
top_terms_stem_stop_tfidf_test


# ### Составление таблицы

# In[37]:


import pandas as pd


# In[38]:


columns = pd.MultiIndex.from_product([['Count', 'TF', 'TF-IDF'], ['Без стоп-
слов', 'С стоп-словами']])


# #### Без стемминга

# In[39]:


df1 = pd.DataFrame(columns=columns)

df1['Count', 'Без стоп-слов'] = top_terms_without_stop
df1['TF', 'Без стоп-слов'] = top_terms_tf
df1['TF-IDF', 'Без стоп-слов'] = top_terms_tfidf

df1['Count', 'С стоп-словами'] = top_terms_stop
df1['TF', 'С стоп-словами'] = top_terms_stop_tf
df1['TF-IDF', 'С стоп-словами'] = top_terms_stop_tfidf

df1


# In[40]:


df2 = pd.DataFrame(columns=columns)

df2['Count', 'Без стоп-слов'] = top_terms_without_stop_test
df2['TF', 'Без стоп-слов'] = top_terms_tf_test
df2['TF-IDF', 'Без стоп-слов'] = top_terms_tfidf_test

df2['Count', 'С стоп-словами'] = top_terms_stop_test
df2['TF', 'С стоп-словами'] = top_terms_stop_tf_test
df2['TF-IDF', 'С стоп-словами'] = top_terms_stop_tfidf_test

df2


# #### Со стеммингом

# In[41]:


df3 = pd.DataFrame(columns=columns)

df3['Count', 'Без стоп-слов'] = top_terms_stem
df3['TF', 'Без стоп-слов'] = top_terms_stem_tf
df3['TF-IDF', 'Без стоп-слов'] = top_terms_stem_tfidf
```

```
df3['Count', 'С стоп-словами'] = top_terms_stop_stem
df3['TF', 'С стоп-словами'] = top_terms_stem_stop_tf
df3['TF-IDF', 'С стоп-словами'] = top_terms_stem_stop_tfidf

df3


# In[42]:


df4 = pd.DataFrame(columns=columns)

df4['Count', 'Без стоп-слов'] = top_terms_stem_test
df4['TF', 'Без стоп-слов'] = top_terms_stem_tf_test
df4['TF-IDF', 'Без стоп-слов'] = top_terms_stem_tfidf_test

df4['Count', 'С стоп-словами'] = top_terms_stop_stem_test
df4['TF', 'С стоп-словами'] = top_terms_stem_stop_tf_test
df4['TF-IDF', 'С стоп-словами'] = top_terms_stem_stop_tfidf_test

df4


# #### Запись в файл

# In[43]:


import openpyxl


# In[44]:


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

df1.to_excel(writer, sheet_name='Train, wo stem')
df2.to_excel(writer, sheet_name='Test, wo stem')
df3.to_excel(writer, sheet_name='Train, with stem')
df4.to_excel(writer, sheet_name='Test, with stem')

writer.close()


# ### Конвейер

# In[45]:


from sklearn.metrics import classification_report
from sklearn.naive_bayes import MultinomialNB


# In[46]:


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


# In[47]:
```

```
def prepare(data, max_feature, stop_word, use_tf, use_idf):
    tf = None
    cv = CountVectorizer(max_features=max_feature, stop_words=stop_word)
    cv.fit(data)
    if use_tf:
        tf = TfidfTransformer(use_idf=use_idf)
        tf.fit(cv.transform(data))
    return cv, tf


# In[48]:


result = []

for max_features_value in max_features_values:
    for stop_word in stop_words:
        for ut in use_tf:
            for ui in use_idf:
                options = {}
                cv, tf = prepare(twenty_train_full.data, max_features_value,
stop_word, ut, ui)
                if tf:
                    clf = MultinomialNB()

clf.fit(tf.transform(cv.transform(twenty_train_full.data)),
twenty_train_full.target)
                    prep_test =
tf.transform(cv.transform(twenty_test_full.data))
                else:
                    clf = MultinomialNB()
                    clf.fit(cv.transform(twenty_train_full.data),
twenty_train_full.target)
                    prep_test = cv.transform(twenty_test_full.data)

                options['features'] = max_features_value
                options['stop_words'] = stop_word
                options['use_tf'] = ut
                options['use_idf'] = ui

                result_data = classification_report(clf.predict(prep_test),
twenty_test_full.target, output_dict=True)
                result_df = pd.DataFrame(result_data)
                result.append({
                    'df': result_df,
                    'options': options
                })


# In[49]:


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

df = pd.DataFrame(columns=['Номер страницы', 'features', 'stop_words',
'use_tf', 'use_idf'])
for it, item in enumerate(result):
    for key, value in item['options'].items():
        df.at[it, key] = value
    df.at[it, 'Номер страницы'] = it

df.to_excel(writer, sheet_name='Оглавление')
```

```
for it, item in enumerate(result):
    df_new = pd.DataFrame(item['df'])
    df_new.to_excel(writer, sheet_name=f'Страница {it}')

writer.close()


# In[50]:


from sklearn.pipeline import Pipeline

parameters = {
    'vect__max_features': max_features_values,
    'vect__stop_words': stop_words,
    'tfidf__use_idf': use_idf
}

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


# In[51]:


from sklearn.model_selection import GridSearchCV

gscv = GridSearchCV(text_clf, param_grid=parameters)
gscv.fit(twenty_train_full.data, twenty_train_full.target)


# In[52]:


print(classification_report(gscv.predict(twenty_test_full.data),
twenty_test_full.target))


# In[53]:


gscv.best_params_
```