Липецкий государственный технический университет

Факультет автоматизации и информатики

Кафедра автоматизированных систем управления

Лабораторная работа № 3

По дисциплине: «Прикладные интеллектуальные системы и экспертные системы**»**

Предварительная обработка текстовых данных

Студент Александрук А.М.

Группа М-ИАП-23-1

Руководитель Кургасов В.В.
к.т.н. доцент

Липецк 2023г.

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 1

Классы: 2,3,8;

Название класса: [comp.graphics, comp.os.ms-windows.misc, rec.autos]

Ход работы:

Подключаем все возможные библиотеки, которые потребуется для выполнения лабораторной работы. Все данные лабораторной работы и ее выполнение рассматриваются на рисунках. Загружаем данные dataset в наш код.

```
# Лабораторная работа 3
# Вариант 1 (классы 2,3,8: comp.graphics, comp.os.ms-windows.misc, rec.autos)
# Подключение библиотек
import pandas as pd
from sklearn.datasets import fetch_20newsgroups
import nltk
from nltk import word_tokenize
from nltk.stem import PorterStemmer
from sklearn.feature_extraction.text import CountVectorizer, TfidfTransformer
from sklearn.neighbors import KNeighborsClassifier
import numpy as np
from sklearn.naive_bayes import MultinomialNB
from sklearn.pipeline import Pipeline
from sklearn.feature_extraction.text import CountVectorizer, TfidfTransformer
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score
from nltk.stem import PorterStemmer
```

Рисунок 1 – Импорт библиотек

```
[54]: # Выбор классов
categories = ['comp.graphics', 'comp.os.ms-windows.misc', 'rec.autos']
remove = ('headers', 'footers', 'quotes')

twenty_train = fetch_20newsgroups(subset='train', shuffle=True, random_state=42, categories=categories, remove=remove)
twenty_test = fetch_20newsgroups(subset='test', shuffle=True, random_state=42, categories=categories, remove=remove)
print(twenty_train.data[3])
print(twenty_test.data[3])

nltk.download('punkt')
```

Рисунок 2 – Загрузка dataset

После загрузки данных выведем текст сообщений и выведем тестовые и обучающиеся выборки.

```
1954 MG-TF with frame-up restoration in early '70's - a local show winner!
Driven very little and stored inside since then - mostly collected dirt &
dust.  Needs attention to brake cylinders (like all MG-T's) but otherwise
ready to run.  Chrome & paint not fancy but it is mechanically excellent.
The engine, a 1250cc, was completely overhauled by a machine shop.  It is
priced at $12,000.

1953 MG-TD Good shape but hasn't been run since '70's.  Needs engine work,
but no rust and everything is with it including a top, side curtains and
carpet that were new and haven't seen the outdoors since the '70's. $9,500.

1952 MG-TD Basket Case.  I'd call it a parts car, but it's too good
for that.  Everything seems to be there except the tach.  Would make
a good project car or parts car if you insist.  No apparent rust but the
upholstery is a disaster.  Stored inside since the '70's.  The top was
new but now soso.  This one has wire wheels!  Looking for $4,500.

All three cars will be sold "as they stand" with no hassles or haggles.
Time has passed by and it is time to part company.  Prices are negotiable.

Reply via matthews@Oswego.oswego.edu or   U.S. mail to: P. O. Box 1015
                                          315-341-3501   Oswego, NY 13126


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

Рисунок 3 – Вывод текста

К нашим полученным данным применим стемминг и выведем следующий текст после обработки.

```
[55]: # Применить стемминг
def stemn(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

porter_stemmer = PorterStemmer()
stem_train = []
for text in twenty_train.data:
    nltk_tokens = word_tokenize(text)
    line = ''
    for word in nltk_tokens:
        line += ' ' + porter_stemmer.stem(word)
    stem_train.append(line)
print(stem_train[0])

stem_test = stemn(twenty_test.data)
```

```
 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 .
```

Рисунок 4 – Применение стемминга

Векторизация выборки со значением max_features = 1000.

```
[56]: # Векторизация выборки
vect = CountVectorizer(max_features=10000, stop_words='english')
train_data = vect.fit_transform(twenty_train.data)
test_data = vect.transform(twenty_test.data)
x = list(zip(vect.get_feature_names_out(), np.ravel(train_data.sum(axis=0))))

def SortbyTF(inputStr):
    return inputStr[1]

x.sort(key=SortbyTF, reverse=True)
print(x[:10])


def tf(input_str):
    return input_str[1]

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

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

train_data_stop = vect_stop.fit_transform(twenty_train.data)
test_data_stop = vect_stop.transform(twenty_test.data)

top_terms_stop = [{term[0]: term[1]} for term in terms(vect_stop, train_data_stop, 20)]
top_terms_stop2 = [{term[0]: term[1]} for term in terms(vect_stop, test_data_stop, 20)]
```

```
[('ax', 62376), ('max', 4482), ('g9v', 1166), ('b8f', 1111), ('a86', 916), ('pl', 823), ('145', 756), ('windows', 752), ('1d9', 672), ('34u', 549)]
```

Рисунок 5 – Векторизация выборки

Затем повторяем действия предыдущего пункта задания с отсечением стоп слов.

```
[57]: # Повторение предыдущего пункта
vect_nostop = CountVectorizer(max_features=10000)

train_data_nostop_stem = vect_nostop.fit_transform(stem_train)
test_data_nostop_stem = vect_nostop.transform(stem_test)
test_data_nostop_stem
top_terms_stem = [{term[0]: term[1]} for term in terms(vect_nostop, train_data_nostop_stem, 20)]
top_terms_stem_test = [{term[0]: term[1]} for term in terms(vect_nostop, test_data_nostop_stem, 20)]
top_terms_stem_test
```

```
[57]: [{'the': 8381},
 {'to': 4623},
 {'and': 4079},
 {'of': 3453},
 {'is': 2984},
 {'it': 2605},
 {'in': 2511},
 {'for': 2297},
 {'that': 1917},
 {'you': 1913},
 {'on': 1502},
 {'thi': 1404},
 {'with': 1363},
 {'have': 1291},
 {'be': 1276},
 {'do': 1211},
 {'are': 1160},
 {'or': 1152},
 {'imag': 1075},
 {'as': 1015}]
```

Рисунок 6 – Код и вывод векторизации с отсечением stop-слов.

Применим векторизацию через TfdgTransformer с использованием (TF и TF-IDF взвешиваний) со stop-слово и без stop-слово.

```
[58]: # Векторизация через TfidgTransformer с использованием (TF и TF-IDF взвешиваний)
tfidf = TfidfTransformer(use_idf=True).fit(train_data)
train_data_tfidf = tfidf.transform(train_data)

# С использованием стоп-слов
train_data_tf = vect_stop.fit_transform(twenty_train.data)
test_data_tf = vect_stop.transform(twenty_test.data)
test_data_tf
train_data_tfidf = tfidf.fit_transform(train_data)
test_data_tfidf = tfidf.transform(test_data)
test_data_tfidf
top_terms_tf = [{term[0]: term[1]} for term in terms(vect_stop, train_data_tf, 20)]
top_terms_tf_test = [{term[0]: term[1]} for term in terms(vect_stop, test_data_tf, 20)]
top_terms_tf_test
top_terms_tfidf = [{term[0]: term[1]} for term in terms(vect_stop, train_data_tfidf, 20)]
top_terms_tfidf_test = [{term[0]: term[1]} for term in terms(vect_stop, test_data_tfidf, 20)]
top_terms_tfidf_test
```

```
[58]: [{'windows': 30.87483669416885},
 {'know': 22.925242613808344},
 {'like': 22.29949503261906},
 {'does': 19.85892398401261},
 {'file': 19.403135041274997},
 {'car': 19.227902099488265},
 {'just': 18.966181057850275},
 {'don': 18.359355377409916},
 {'thanks': 18.00668668828334},
 {'dos': 17.35514324169421},
 {'graphics': 16.421579988160406},
 {'think': 15.776590789702308},
 {'program': 15.237120123579587},
 {'use': 15.144848169850718},
 {'ve': 14.68659030665054},
 {'problem': 14.541134609419773},
 {'edu': 13.888103325902122},
 {'new': 13.780274911406302},
 {'os': 13.657941301291583},
 {'good': 13.611892242416218}]
```

Рисунок 7 – Код выполнения и вывод результатов с использованием взвешиваний со Stop-слова

```
[59]: # Без стоп-слов
train_data_tf = vect_nostop.fit_transform(twenty_train.data)
test_data_tf = vect_nostop.transform(twenty_test.data)
test_data_tf

train_data_notfidf = tfidf.fit_transform(train_data)
test_data_notfidf = tfidf.transform(test_data)
test_data_notfidf

top_terms_notf = [{term[0]: term[1]} for term in terms(vect_nostop, train_data_tf, 20)]
top_terms_notf_test = [{term[0]: term[1]} for term in terms(vect_nostop, test_data_tf, 20)]
top_terms_notf_test
top_terms_notfidf = [{term[0]: term[1]} for term in terms(vect_nostop, train_data_tfidf, 20)]
top_terms_notfidf_test = [{term[0]: term[1]} for term in terms(vect_nostop, test_data_tfidf, 20)]
top_terms_notfidf_test
```

```
[59]: [{'wilson': 30.87483669416885},
 {'ka': 22.925242613808344},
 {'levels': 22.29949503261906},
 {'djgpp': 19.85892398401261},
 {'fee': 19.403135041274997},
 {'camera': 19.227902099488265},
 {'january': 18.966181057850275},
 {'dll': 18.359355377409916},
 {'telephone': 18.00668668828334},
 {'dm_': 17.35514324169421},
 {'gj': 16.421579988160406},
 {'tercel': 15.776590789702308},
 {'processing': 15.237120123579587},
 {'unit': 15.144848169850718},
 {'v9': 14.68659030665054},
 {'printer': 14.541134609419773},
 {'easiest': 13.888103325902122},
 {'ngl': 13.780274911406302},
 {'orpu': 13.657941301291583},
 {'getting': 13.611892242416218}]
```

Рисунок 8 – Код выполнения и вывод результатов с использованием взвешиваний со Без stop-словом

Используем конвейер (Pipeline), реализуем модель Наивного Байесовского классификатора и выявить на основе показателей качества (значения полноты, точности, f1-меры и аккуратности).

```
[69]: # Модель Наивного Байесовского классификатора
# Определение функций
def stemm(text):
    porter_stemmer = PorterStemmer()
    nltk_tokens = nltk.word_tokenize(text)
    stems = [porter_stemmer.stem(word) for word in nltk_tokens]
    return ' '.join(stems)

# Создание и обучение конвейеров
pipelines = {
    "Стемминг без удаления стоп-слов": Pipeline([
        ('vect', CountVectorizer(analyzer=stemm)),
        ('tfidf', TfidfTransformer(use_idf=True)),
        ('clf', MultinomialNB())
    ]),
    "С отсечением стоп-слов": Pipeline([
        ('vect', CountVectorizer(stop_words='english')),
        ('tfidf', TfidfTransformer(use_idf=True)),
        ('clf', MultinomialNB())
    ]),
    "Стемминг с отсечением стоп-слов": Pipeline([
        ('vect', CountVectorizer(analyzer=stemm, stop_words='english')),
        ('tfidf', TfidfTransformer(use_idf=True)),
        ('clf', MultinomialNB())
    ]),
}

# Оценка конвейеров
results = {}
for name, pipeline in pipelines.items():
    pipeline = pipeline.fit(twenty_train.data, twenty_train.target)
    prediction = pipeline.predict(twenty_test.data)

    accuracy = accuracy_score(twenty_test.target, prediction)
    precision = precision_score(twenty_test.target, prediction, average='weighted')
    recall = recall_score(twenty_test.target, prediction, average='weighted')
    f1 = f1_score(twenty_test.target, prediction, average='weighted')

    results[name] = {"Accuracy": accuracy, "Precision": precision, "Recall": recall, "F1 Score": f1}

# Вывод результатов
for name, metrics in results.items():
    print(f"Метрики производительности для {name}:")
    print(f"  Accuracy: {metrics['Accuracy']:.4f}")
    print(f"  Precision: {metrics['Precision']:.4f}")
    print(f"  Recall: {metrics['Recall']:.4f}")
    print(f"  F1 Score: {metrics['F1 Score']:.4f}")
    print()
```

Рисунок 9 – Код выполнения реализации модели Наивного Байесовского классификатора

```
Метрики производительности для Стемминг без удаления стоп-слов:
  Accuracy: 0.4283
  Precision: 0.4710
  Recall: 0.4283
  F1 Score: 0.3919

Метрики производительности для С отсечением стоп-слов:
  Accuracy: 0.8702
  Precision: 0.8703
  Recall: 0.8702
  F1 Score: 0.8686

Метрики производительности для Стемминг с отсечением стоп-слов:
  Accuracy: 0.4283
  Precision: 0.4710
  Recall: 0.4283
  F1 Score: 0.3919
```

Рисунок 10 – Результат работы модели Наивного Байесовского классификатора

По результатам 6 пункта построим 4 таблицы по выборки.

Таблица 1 – Результат векторизации данных обучающего множества без стемминга

	Count		TF		TF-IDF	
№	Без стоп-слов	С стоп-словами	Без стоп-слов	С стоп-словами	Без стоп-слов	С стоп-словами
0	{'ax': 62376}	{'image': 713}	{'ax': 62376}	{'ax': 62376}	{'windows': 42.19886950770242}	{'windows': 42.19886950770242}
1	{'max': 4482}	{'windows': 623}	{'the': 10022}	{'max': 4482}	{'car': 30.82517993716325}	{'car': 30.82517993716325}
2	{'g9v': 1166}	{'file': 613}	{'to': 5235}	{'g9v': 1166}	{'know': 30.64641671574011}	{'know': 30.64641671574011}
3	{'b8f': 1111}	{'jpeg': 526}	{'max': 4482}	{'b8f': 1111}	{'thanks': 30.259756602254114}	{'thanks': 30.259756602254114}
4	{'a86': 916}	{'edu': 502}	{'and': 4441}	{'a86': 916}	{'like': 29.81950675014476}	{'like': 29.81950675014476}
5	{'pl': 823}	{'graphics': 489}	{'of': 3910}	{'pl': 823}	{'just': 27.32832770102855}	{'just': 27.32832770102855}
6	{'145': 756}	{'like': 442}	{'is': 3414}	{'145': 756}	{'file': 26.95279520647367}	{'file': 26.95279520647367}
7	{'windows': 752}	{'use': 408}	{'it': 3066}	{'windows': 752}	{'does': 26.22569651463021}	{'does': 26.22569651463021}
8	{'1d9': 672}	{'know': 367}	{'in': 2877}	{'1d9': 672}	{'use': 25.479266177864712}	{'use': 25.479266177864712}
9	{'34u': 549}	{'don': 364}	{'for': 2655}	{'34u': 549}	{'files': 23.35081949234985}	{'files': 23.35081949234985}
10	{'use': 533}	{'car': 349}	{'that': 2312}	{'use': 533}	{'don': 23.212614450110554}	{'don': 23.212614450110554}
11	{'like': 527}	{'images': 340}	{'you': 2282}	{'like': 527}	{'good': 21.350194940694724}	{'good': 21.350194940694724}
12	{'file': 519}	{'available': 337}	{'on': 1818}	{'file': 519}	{'problem': 21.346823318643455}	{'problem': 21.346823318643455}
13	{'car': 515}	{'files': 331}	{'have': 1600}	{'car': 515}	{'graphics': 20.542700306957354}	{'graphics': 20.542700306957354}
14	{'1t': 510}	{'just': 329}	{'this': 1563}	{'1t': 510}	{'program': 20.49420240465935}	{'program': 20.49420240465935}
15	{'0t': 505}	{'does': 322}	{'with': 1522}	{'0t': 505}	{'think': 20.034205878616042}	{'think': 20.034205878616042}
16	{'image': 505}	{'data': 313}	{'or': 1365}	{'image': 505}	{'need': 19.37256987228952}	{'need': 19.37256987228952}
17	{'bhj': 456}	{'ftp': 312}	{'be': 1351}	{'bhj': 456}	{'using': 18.994986442384565}	{'using': 18.994986442384565}
18	{'just': 455}	{'bit': 310}	{'are': 1323}	{'just': 455}	{'ve': 18.908332905355316}	{'ve': 18.908332905355316}
19	{'edu': 453}	{'software': 306}	{'as': 1309}	{'edu': 453}	{'dos': 18.848167962055555}	{'dos': 18.848167962055555}

Таблица 2 – Результат векторизации данных тестового множества без стемминга

	Count		TF		TF-IDF	
№	**Без стоп-слов**	**С стоп-словами**	**Без стоп-слов**	**С стоп-словами**	**Без стоп-слов**	**С стоп-словами**
0	{'ax': 62376}	{'ax': 62376}	{'image': 713}	{'image': 713}	{'windows': 30.87483669416885}	{'wilson': 30.87483669416885}
1	{'the': 10019}	{'max': 4482}	{'windows': 623}	{'windows': 623}	{'know': 22.925242613808344}	{'ka': 22.925242613808344}
2	{'to': 5239}	{'g9v': 1166}	{'file': 613}	{'file': 613}	{'like': 22.29949503261906}	{'levels': 22.29949503261906}
3	{'max': 4482}	{'b8f': 1111}	{'jpeg': 526}	{'jpeg': 526}	{'does': 19.85892398401261}	{'djgpp': 19.85892398401261}
4	{'and': 4441}	{'a86': 916}	{'edu': 502}	{'edu': 502}	{'file': 19.403135041274997}	{'fee': 19.403135041274997}
5	{'of': 3911}	{'pl': 823}	{'graphics': 489}	{'graphics': 489}	{'car': 19.227902099488265}	{'camera': 19.227902099488265}
6	{'is': 3442}	{'145': 756}	{'like': 442}	{'like': 442}	{'just': 18.966181057850275}	{'january': 18.966181057850275}
7	{'it': 3264}	{'windows': 752}	{'use': 408}	{'use': 408}	{'don': 18.359355377409916}	{'dll': 18.359355377409916}
8	{'in': 2883}	{'1d9': 672}	{'know': 367}	{'know': 367}	{'thanks': 18.00668668828334}	{'telephone': 18.00668668828334}
9	{'for': 2655}	{'34u': 549}	{'don': 364}	{'don': 364}	{'dos': 17.35514324169421}	{'dm_': 17.35514324169421}
10	{'that': 2320}	{'use': 533}	{'car': 349}	{'car': 349}	{'graphics': 16.421579988160406}	{'gj': 16.421579988160406}
11	{'you': 2282}	{'like': 527}	{'images': 340}	{'images': 340}	{'think': 15.776590789702308}	{'tercel': 15.776590789702308}
12	{'on': 1822}	{'file': 519}	{'available': 337}	{'available': 337}	{'program': 15.237120123579587}	{'processing': 15.237120123579587}
13	{'have': 1729}	{'car': 515}	{'files': 331}	{'files': 331}	{'use': 15.144848169850718}	{'unit': 15.144848169850718}
14	{'thi': 1564}	{'1t': 510}	{'just': 329}	{'just': 329}	{'ve': 14.68659030665054}	{'v9': 14.68659030665054}
15	{'with': 1523}	{'0t': 505}	{'does': 322}	{'does': 322}	{'problem': 14.541134609419773}	{'printer': 14.541134609419773}
16	{'be': 1446}	{'image': 505}	{'data': 313}	{'data': 313}	{'edu': 13.888103325902122}	{'easiest': 13.888103325902122}
17	{'or': 1369}	{'bhj': 456}	{'ftp': 312}	{'ftp': 312}	{'new': 13.780274911406302}	{'ng1': 13.780274911406302}
18	{'do': 1364}	{'just': 455}	{'bit': 310}	{'bit': 310}	{'os': 13.657941301291583}	{'orpu': 13.657941301291583}
19	{'are': 1355}	{'edu': 453}	{'software': 306}	{'software': 306}	{'good': 13.611892242416218}	{'getting': 13.611892242416218}

Таблица 3 – Результат векторизации данных обучающего множества со стеммигом

	Count		TF		TF-IDF	
№	**Без стоп-слов**	**С стоп-словами**	**Без стоп-слов**	**С стоп-словами**	**Без стоп-слов**	**С стоп-словами**
0	{'ax': 62376}	{'ax': 62376}	{'the': 8381}	{'ax': 62376}	{'wilson': 30.87483669416885}	{'wilson': 42.19886950770242}
1	{'the': 10019}	{'max': 4482}	{'to': 4623}	{'max': 4482}	{'ka': 22.925242613808344}	{'camera': 30.82517993716325}
2	{'to': 5239}	{'g9v': 1166}	{'and': 4079}	{'g9v': 1166}	{'levels': 22.29949503261906}	{'ka': 30.64641671574011}
3	{'max': 4482}	{'b8f': 1111}	{'of': 3453}	{'b8f': 1111}	{'djgpp': 19.85892398401261}	{'telephone': 30.259756602254114}
4	{'and': 4441}	{'a86': 916}	{'is': 2984}	{'a86': 916}	{'fee': 19.403135041274997}	{'levels': 29.81950675014476}
5	{'of': 3911}	{'pl': 823}	{'it': 2605}	{'pl': 823}	{'camera': 19.227902099488265}	{'january': 27.32832770102855}
6	{'is': 3442}	{'145': 756}	{'in': 2511}	{'145': 756}	{'january': 18.966181057850275}	{'fee': 26.95279520647367}
7	{'it': 3264}	{'windows': 752}	{'for': 2297}	{'windows': 752}	{'dll': 18.359355377409916}	{'djgpp': 26.22569651463021}
8	{'in': 2883}	{'1d9': 672}	{'that': 1917}	{'1d9': 672}	{'telephone': 18.00668668828334}	{'unit': 25.479266177864712}
9	{'for': 2655}	{'34u': 549}	{'you': 1913}	{'34u': 549}	{'dm_': 17.35514324169421}	{'feeling': 23.35081949234985}
10	{'that': 2320}	{'use': 533}	{'on': 1502}	{'use': 533}	{'gj': 16.421579988160406}	{'dll': 23.212614450110554}
11	{'you': 2282}	{'like': 527}	{'thi': 1404}	{'like': 527}	{'tercel': 15.776590789702308}	{'getting': 21.350194940694724}
12	{'on': 1822}	{'file': 519}	{'with': 1363}	{'file': 519}	{'processing': 15.237120123579587}	{'printer': 21.346823318643455}
13	{'have': 1729}	{'car': 515}	{'have': 1291}	{'car': 515}	{'unit': 15.144848169850718}	{'gj': 20.542700306957354}
14	{'thi': 1564}	{'1t': 510}	{'be': 1276}	{'1t': 510}	{'v9': 14.68659030665054}	{'processing': 20.49420240465935}
15	{'with': 1523}	{'0t': 505}	{'do': 1211}	{'0t': 505}	{'printer': 14.541134609419773}	{'tercel': 20.034205878616042}
16	{'be': 1446}	{'image': 505}	{'are': 1160}	{'image': 505}	{'easiest': 13.888103325902122}	{'network': 19.37256987228952}
17	{'or': 1369}	{'bhj': 456}	{'or': 1152}	{'bhj': 456}	{'ng1': 13.780274911406302}	{'unknown': 18.994986442384565}
18	{'do': 1364}	{'just': 455}	{'imag': 1075}	{'just': 455}	{'orpu': 13.657941301291583}	{'v9': 18.908332905355316}
19	{'are': 1355}	{'edu': 453}	{'as': 1015}	{'edu': 453}	{'getting': 13.611892242416218}	{'dm_': 18.848167962055555}

Таблица 3 – Результат векторизации данных тестового множества со стеммигом

	Count		TF		TF-IDF	
№	**Без стоп-слов**	**С стоп-словами**	**Без стоп-слов**	**С стоп-словами**	**Без стоп-слов**	**С стоп-словами**
0	{'the': 8381}	{'the': 8381}	{'image': 713}	{'windows': 30.87483669416885}	{'windows': 30.87483669416885}	{'wilson': 30.87483669416885}
1	{'to': 4623}	{'to': 4623}	{'windows': 623}	{'know': 22.925242613808344}	{'know': 22.925242613808344}	{'ka': 22.925242613808344}
2	{'and': 4079}	{'and': 4079}	{'file': 613}	{'like': 22.29949503261906}	{'like': 22.29949503261906}	{'levels': 22.29949503261906}
3	{'of': 3453}	{'of': 3453}	{'jpeg': 526}	{'does': 19.85892398401261}	{'does': 19.85892398401261}	{'djgpp': 19.85892398401261}
4	{'is': 2984}	{'is': 2984}	{'edu': 502}	{'file': 19.403135041274997}	{'file': 19.403135041274997}	{'fee': 19.403135041274997}
5	{'it': 2605}	{'it': 2605}	{'graphics': 489}	{'car': 19.227902099488265}	{'car': 19.227902099488265}	{'camera': 19.227902099488265}
6	{'in': 2511}	{'in': 2511}	{'like': 442}	{'just': 18.966181057850275}	{'just': 18.966181057850275}	{'january': 18.966181057850275}
7	{'for': 2297}	{'for': 2297}	{'use': 408}	{'don': 18.359355377409916}	{'don': 18.359355377409916}	{'dll': 18.359355377409916}
8	{'that': 1917}	{'that': 1917}	{'know': 367}	{'thanks': 18.00668668828334}	{'thanks': 18.00668668828334}	{'telephone': 18.00668668828334}
9	{'you': 1913}	{'you': 1913}	{'don': 364}	{'dos': 17.35514324169421}	{'dos': 17.35514324169421}	{'dm_': 17.35514324169421}
10	{'on': 1502}	{'on': 1502}	{'car': 349}	{'graphics': 16.421579988160406}	{'graphics': 16.421579988160406}	{'gj': 16.421579988160406}
11	{'thi': 1404}	{'thi': 1404}	{'images': 340}	{'think': 15.776590789702308}	{'think': 15.776590789702308}	{'tercel': 15.776590789702308}
12	{'with': 1363}	{'with': 1363}	{'available': 337}	{'program': 15.237120123579587}	{'program': 15.237120123579587}	{'processing': 15.237120123579587}
13	{'have': 1291}	{'have': 1291}	{'files': 331}	{'use': 15.144848169850718}	{'use': 15.144848169850718}	{'unit': 15.144848169850718}
14	{'be': 1276}	{'be': 1276}	{'just': 329}	{'ve': 14.68659030665054}	{'ve': 14.68659030665054}	{'v9': 14.68659030665054}
15	{'do': 1211}	{'do': 1211}	{'does': 322}	{'problem': 14.541134609419773}	{'problem': 14.541134609419773}	{'printer': 14.541134609419773}
16	{'are': 1160}	{'are': 1160}	{'data': 313}	{'edu': 13.888103325902122}	{'edu': 13.888103325902122}	{'easiest': 13.888103325902122}
17	{'or': 1152}	{'or': 1152}	{'ftp': 312}	{'new': 13.780274911406302}	{'new': 13.780274911406302}	{'ng1': 13.780274911406302}
18	{'imag': 1075}	{'imag': 1075}	{'bit': 310}	{'os': 13.657941301291583}	{'os': 13.657941301291583}	{'orpu': 13.657941301291583}
19	{'as': 1015}	{'as': 1015}	{'software': 306}	{'good': 13.611892242416218}	{'good': 13.611892242416218}	{'getting': 13.611892242416218}

Код программы: https://github.com/lipadirka/Prikladnie-intelektualn-systems.git

Вывод

В ходе выполнения лабораторной работы в процессе изучения принципов обработки текстовых данных, мы обнаружили, что лучший способ предварительной обработки данных — это векторизация TfidfTransformer с использованием TF-IDF.