Московский государственный технический университет им. Н.Э. Баумана
Факультет «Информатика и системы управления»
Кафедра «Системы обработки информации и управления»

Лабораторная работа №5
по курсу «Методы машинного обучения»

«Предобработка и классификация текста»

ИСПОЛНИТЕЛЬ:

Ерохин И.А.
Группа ИУ5-24М

_____

"__"_____2022 г.

Москва 2022

In [39]:
```
text = 'Фотографирован утром. Счастливо лает «абыр», повторяя это слово громко и как б
text2 = 'Какая гадина, а ещё пролетарий. Господи, боже мой – как больно! До костей про
```

In [40]:
```
!pip install natasha
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/pu
blic/simple/
Requirement already satisfied: natasha in /usr/local/lib/python3.7/dist-packages (1.4.
0)
Requirement already satisfied: razdel>=0.5.0 in /usr/local/lib/python3.7/dist-packages
(from natasha) (0.5.0)
Requirement already satisfied: pymorphy2 in /usr/local/lib/python3.7/dist-packages (fr
om natasha) (0.9.1)
Requirement already satisfied: navec>=0.9.0 in /usr/local/lib/python3.7/dist-packages
(from natasha) (0.10.0)
Requirement already satisfied: yargy>=0.14.0 in /usr/local/lib/python3.7/dist-packages
(from natasha) (0.15.0)
Requirement already satisfied: ipymarkup>=0.8.0 in /usr/local/lib/python3.7/dist-packa
ges (from natasha) (0.9.0)
Requirement already satisfied: slovnet>=0.3.0 in /usr/local/lib/python3.7/dist-package
s (from natasha) (0.5.0)
Requirement already satisfied: intervaltree>=3 in /usr/local/lib/python3.7/dist-packag
es (from ipymarkup>=0.8.0->natasha) (3.1.0)
Requirement already satisfied: sortedcontainers<3.0,>=2.0 in /usr/local/lib/python3.7/
dist-packages (from intervaltree>=3->ipymarkup>=0.8.0->natasha) (2.4.0)
Requirement already satisfied: numpy in /usr/local/lib/python3.7/dist-packages (from n
avec>=0.9.0->natasha) (1.21.6)
Requirement already satisfied: docopt>=0.6 in /usr/local/lib/python3.7/dist-packages
(from pymorphy2->natasha) (0.6.2)
Requirement already satisfied: dawg-python>=0.7.1 in /usr/local/lib/python3.7/dist-pac
kages (from pymorphy2->natasha) (0.7.2)
Requirement already satisfied: pymorphy2-dicts-ru<3.0,>=2.4 in /usr/local/lib/python3.
7/dist-packages (from pymorphy2->natasha) (2.4.417127.4579844)
```

Задача токенизации

In [41]:
```
from razdel import tokenize, sentenize
```

In [42]:
```
n_tok_text = list(tokenize(text))
n_tok_text
```

Out[42]:
```
[Substring(0, 14, 'Фотографирован'),
 Substring(15, 20, 'утром'),
 Substring(20, 21, '.'),
 Substring(22, 31, 'Счастливо'),
 Substring(32, 36, 'лает'),
 Substring(37, 38, '«'),
 Substring(38, 42, 'абыр'),
 Substring(42, 43, '»'),
 Substring(43, 44, ','),
 Substring(45, 53, 'повторяя'),
 Substring(54, 57, 'это'),
 Substring(58, 63, 'слово'),
 Substring(64, 70, 'громко'),
 Substring(71, 72, 'и'),
 Substring(73, 76, 'как'),
 Substring(77, 79, 'бы'),
 Substring(80, 88, 'радостно'),
 Substring(88, 89, '.'),
 Substring(90, 91, 'В'),
 Substring(92, 93, '3'),
 Substring(94, 98, 'часа'),
 Substring(99, 102, 'дня'),
 Substring(103, 104, '('),
 Substring(104, 112, 'крупными'),
 Substring(113, 120, 'буквами'),
 Substring(120, 121, ')'),
 Substring(122, 131, 'засмеялся'),
 Substring(131, 132, ','),
 Substring(133, 139, 'вызвав'),
 Substring(140, 147, 'обморок'),
 Substring(148, 157, 'горничной'),
 Substring(158, 162, 'Зины'),
 Substring(162, 163, '.'),
 Substring(164, 171, 'Вечером'),
 Substring(172, 180, 'произнёс'),
 Substring(181, 182, '8'),
 Substring(183, 186, 'раз'),
 Substring(187, 193, 'подряд'),
 Substring(194, 199, 'слово'),
 Substring(200, 201, '«'),
 Substring(201, 210, 'абыр-валг'),
 Substring(210, 211, '»'),
 Substring(211, 212, ','),
 Substring(213, 214, '«'),
 Substring(214, 218, 'абыр'),
 Substring(218, 219, '»'),
 Substring(219, 220, '.')]
```

In [43]:
```
[_.text for _ in n_tok_text]
```

Out[43]:
```
['Фотографирован',
 'утром',
 '.',
 'Счастливо',
 'лает',
 '«',
 'абыр',
 '»',
 ',',
 'повторяя',
 'это',
 'слово',
 'громко',
 'и',
 'как',
 'бы',
 'радостно',
 '.',
 'В',
 '3',
 'часа',
 'дня',
 '(',
 'крупными',
 'буквами',
 ')',
 'засмеялся',
 ',',
 'вызвав',
 'обморок',
 'горничной',
 'Зины',
 '.',
 'Вечером',
 'произнёс',
 '8',
 'раз',
 'подряд',
 'слово',
 '«',
 'абыр-валг',
 '»',
 ',',
 '«',
 'абыр',
 '»',
 '.']
```

In [44]:
```
n_sen_text = list(sentenize(text))
n_sen_text
```

Out[44]:
```
[Substring(0, 21, 'Фотографирован утром.'),
 Substring(22,
           89,
           'Счастливо лает «абыр», повторяя это слово громко и как бы радостно.'),
 Substring(90,
           163,
           'В 3 часа дня (крупными буквами) засмеялся, вызвав обморок горничной Зин
ы.'),
 Substring(164,
           220,
           'Вечером произнёс 8 раз подряд слово «абыр-валг», «абыр».')]
```

In [45]:
```
[_.text for _ in n_sen_text], len([_.text for _ in n_sen_text])
```

Out[45]:
```
(['Фотографирован утром.',
  'Счастливо лает «абыр», повторяя это слово громко и как бы радостно.',
  'В 3 часа дня (крупными буквами) засмеялся, вызвав обморок горничной Зины.',
  'Вечером произнёс 8 раз подряд слово «абыр-валг», «абыр».'],
 4)
```

In [46]:
```
# Этот вариант токенизации нужен для последующей обработки
def n_sentenize(text):
    n_sen_chunk = []
    for sent in sentenize(text):
        tokens = [_.text for _ in tokenize(sent.text)]
        n_sen_chunk.append(tokens)
    return n_sen_chunk
```

In [47]:
```
n_sen_chunk = n_sentenize(text)
n_sen_chunk
```

Out[47]:
```
[['Фотографирован', 'утром', '.'],
 ['Счастливо',
  'лает',
  '«',
  'абыр',
  '»',
  ',',
  'повторяя',
  'это',
  'слово',
  'громко',
  'и',
  'как',
  'бы',
  'радостно',
  '.'],
 ['В',
  '3',
  'часа',
  'дня',
  '(',
  'крупными',
  'буквами',
  ')',
  'засмеялся',
  ',',
  'вызвав',
  'обморок',
  'горничной',
  'Зины',
  '.'],
 ['Вечером',
  'произнёс',
  '8',
  'раз',
  'подряд',
  'слово',
  '«',
  'абыр-валг',
  '»',
  ',',
  '«',
  'абыр',
  '»',
  '.']]
```

In [48]:
```
n_sen_chunk_2 = n_sentenize(text2)
n_sen_chunk_2
```

Out[48]:
```
[['Какая', 'гадина', ',', 'а', 'ещё', 'пролетарий', '.'],
 ['Господи', ',', 'боже', 'мой', '–', 'как', 'больно', '!'],
 ['До', 'костей', 'проело', 'кипяточком', '.'],
 ['Я',
  'теперь',
  'вою',
  ',',
  'вою',
  ',',
  'да',
  'разве',
  'воем',
  'поможешь',
  '.']]
```

Частеречная разметка

In [49]:
```
from navec import Navec
from slovnet import Morph
```

In [50]:
```
from google.colab import drive
drive.mount('/content/drive')
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mo
unt("/content/drive", force_remount=True).
```

In [51]:
```
# https://github.com/natasha/navec#downloads
navec = Navec.load('/content/drive/MyDrive/Colab Notebooks/navec_news_v1_1B_250K_300d_
```

In [52]:
```
# https://github.com/natasha/slovnet#downloads
n_morph = Morph.load('/content/drive/MyDrive/Colab Notebooks/slovnet_ner_news_v1.tar',
```

In [53]:
```
morph_res = n_morph.navec(navec)
```

In [54]:
```
def print_pos(markup):
    for token in markup.tokens:
        print('{} - {}'.format(token.text, token.tag))
```

In [55]:
```
n_text_markup = list(_ for _ in n_morph.map(n_sen_chunk))
[print_pos(x) for x in n_text_markup]
```

```
Фотографирован - B-PER
утром - O
. - O
Счастливо - B-ORG
лает - O
« - O
абыр - O
» - O
, - O
повторяя - O
это - O
слово - O
громко - O
и - O
как - O
бы - O
радостно - O
. - O
В - O
3 - O
часа - O
дня - O
( - O
крупными - O
буквами - O
) - O
засмеялся - O
, - O
вызвав - O
обморок - O
горничной - O
Зины - B-PER
. - O
Вечером - O
произнёс - O
8 - O
раз - O
подряд - O
слово - O
« - O
абыр-валг - O
» - O
, - O
« - O
абыр - O
» - O
. - O
```

Out[55]:
```
[None, None, None, None]
```

In [56]:
```
n_text2_markup = list(n_morph.map(n_sen_chunk_2))
[print_pos(x) for x in n_text2_markup]
```

```
Какая - O
гадина - O
, - O
а - O
ещё - O
пролетарий - O
. - O
Господи - O
, - O
боже - O
мой - O
– - O
как - O
больно - O
! - O
До - O
костей - O
проело - O
кипяточком - O
. - O
Я - O
теперь - O
вою - O
, - O
вою - O
, - O
да - O
разве - O
воем - O
поможешь - O
. - O
```

Out[56]:
```
[None, None, None, None]
```

Лемматизация

In [57]:
```
from natasha import Doc, Segmenter, NewsEmbedding, NewsMorphTagger, MorphVocab
```

In [58]:
```
def n_lemmatize(text):
    emb = NewsEmbedding()
    morph_tagger = NewsMorphTagger(emb)
    segmenter = Segmenter()
    morph_vocab = MorphVocab()
    doc = Doc(text)
    doc.segment(segmenter)
    doc.tag_morph(morph_tagger)
    for token in doc.tokens:
        token.lemmatize(morph_vocab)
    return doc
```

In [59]:
```
n_doc = n_lemmatize(text)
{_.text: _.lemma for _ in n_doc.tokens}
```

Out[59]:
```
{'(': '(',
 ')': ')',
 ',': ',',
 '.': '.',
 '3': '3',
 '8': '8',
 '«': '«',
 '»': '»',
 'В': 'в',
 'Вечером': 'вечер',
 'Зины': 'зина',
 'Счастливо': 'счастливо',
 'Фотографирован': 'фотографировать',
 'абыр': 'абыр',
 'абыр-валг': 'абыр-валга',
 'буквами': 'буква',
 'бы': 'бы',
 'вызвав': 'вызвать',
 'горничной': 'горничный',
 'громко': 'громко',
 'дня': 'день',
 'засмеялся': 'засмеяться',
 'и': 'и',
 'как': 'как',
 'крупными': 'крупный',
 'лает': 'лаять',
 'обморок': 'обморок',
 'повторяя': 'повторять',
 'подряд': 'подряд',
 'произнёс': 'произнести',
 'радостно': 'радостно',
 'раз': 'раз',
 'слово': 'слово',
 'утром': 'утро',
 'часа': 'час',
 'это': 'это'}
```

In [60]:
```
n_doc2 = n_lemmatize(text2)
{_.text: _.lemma for _ in n_doc2.tokens}
```

Out[60]:
```
{'!': '!',
 ',': ',',
 '.': '.',
 'Господи': 'господь',
 'До': 'до',
 'Какая': 'какой',
 'Я': 'я',
 'а': 'а',
 'боже': 'бог',
 'больно': 'больно',
 'воем': 'вой',
 'вою': 'вой',
 'гадина': 'гадина',
 'да': 'да',
 'ещё': 'еще',
 'как': 'как',
 'кипяточком': 'кипяточек',
 'костей': 'кость',
 'мой': 'мой',
 'поможешь': 'помочь',
 'проело': 'проесть',
 'пролетарий': 'пролетарий',
 'разве': 'разве',
 'теперь': 'теперь',
 '–': '–'}
```

Выделение (распознавание) именованных сущностей

In [61]:
```
from slovnet import NER
from ipymarkup import show_span_ascii_markup as show_markup
```

In [62]:
```
ner = NER.load('/content/drive/MyDrive/Colab Notebooks/slovnet_ner_news_v1.tar')
```

In [63]:
```
ner_res = ner.navec(navec)
```

In [64]:
```
markup_ner = ner(text2)
markup_ner
```

Out[64]:
```
SpanMarkup(
    text='Какая гадина, а ещё пролетарий. Господи, боже мой – как больно! До костей пр
оело кипяточком. Я теперь вою, вою, да разве воем поможешь.',
    spans=[]
)
```

In [65]:
```
show_markup(markup_ner.text, markup_ner.spans)
```

```
Какая гадина, а ещё пролетарий. Господи, боже мой – как больно! До
костей проело кипяточком. Я теперь вою, вою, да разве воем поможешь.
```

Разбор предложения

In [66]:
```
from natasha import NewsSyntaxParser
```

In [67]:
```
emb = NewsEmbedding()
syntax_parser = NewsSyntaxParser(emb)
```

In [71]:
```
n_doc.parse_syntax(syntax_parser)
n_doc.sents[1].syntax.print()
```

```
Счастливо advmod
лает
«         punct
абыр      obl
»         punct
,         punct
повторяя  advcl
это       det
слово     obj
громко    advmod
и         cc
как       advmod
бы        fixed
радостно  advmod
.         punct
```

In [72]:
```
n_doc.parse_syntax(syntax_parser)
n_doc.sents[2].syntax.print()
```

```
В         case
3         nummod
часа      obl
дня       nmod
(         punct
крупными  amod
буквами   parataxis
)         punct
засмеялся
,         punct
вызвав    advcl
обморок   obj
горничной nmod
Зины      nmod
.         punct
```

In [73]:
```
n_doc.parse_syntax(syntax_parser)
n_doc.sents[3].syntax.print()
```

```
Вечером   obl
произнёс
8         nummod
раз       obl
подряд    advmod
слово     nsubj
«         punct
абыр-валг appos
»         punct
,         punct
«         punct
абыр      conj
»         punct
.         punct
```

In [74]:
```
n_doc2.parse_syntax(syntax_parser)
n_doc2.sents[0].syntax.print()
```

```
Какая      det
гадина
,          punct
а          cc
ещё        advmod
пролетарий conj
.          punct
```

In [75]:
```
n_doc2.parse_syntax(syntax_parser)
n_doc2.sents[1].syntax.print()
```

```
Господи parataxis
,       punct
боже
мой     nsubj
–       punct
как     advmod
больно  orphan
!       punct
```

In [76]:
```
import numpy as np
import pandas as pd
from typing import Dict, Tuple
from sklearn.feature_extraction.text import CountVectorizer, TfidfVectorizer
from sklearn.model_selection import GridSearchCV, RandomizedSearchCV
from sklearn.metrics import accuracy_score, balanced_accuracy_score
from sklearn.metrics import precision_score, recall_score, f1_score, classification_re
from sklearn.metrics import confusion_matrix
from sklearn.model_selection import cross_val_score
from sklearn.pipeline import Pipeline
from sklearn.metrics import mean_absolute_error, mean_squared_error, mean_squared_log_
from sklearn.metrics import roc_curve, roc_auc_score
from sklearn.naive_bayes import MultinomialNB
from sklearn.linear_model import LogisticRegression
from sklearn.neighbors import KNeighborsClassifier
from sklearn.svm import LinearSVC
from sklearn.model_selection import train_test_split
import seaborn as sns
from collections import Counter
from sklearn.datasets import fetch_20newsgroups
import matplotlib.pyplot as plt

%matplotlib inline
sns.set(style="ticks")
```

Векторизация текста на основе модели "мешка слов"

In [77]:
```
categories = ["rec.sport.hockey", "sci.electronics", "sci.med"]
newsgroups = fetch_20newsgroups(subset='train', categories=categories)
data = newsgroups['data']
```

In [78]:
```
def accuracy_score_for_classes(
    y_true: np.ndarray,
    y_pred: np.ndarray) -> Dict[int, float]:
    """
    Вычисление метрики accuracy для каждого класса
    y_true - истинные значения классов
    y_pred - предсказанные значения классов
    Возвращает словарь: ключ - метка класса,
    значение - Accuracy для данного класса
    """
    # Для удобства фильтрации сформируем Pandas DataFrame
    d = {'t': y_true, 'p': y_pred}
    df = pd.DataFrame(data=d)
    # Метки классов
    classes = np.unique(y_true)
    # Результирующий словарь
    res = dict()
    # Перебор меток классов
    for c in classes:
        # отфильтруем данные, которые соответствуют
        # текущей метке класса в истинных значениях
        temp_data_flt = df[df['t']==c]
        # расчет accuracy для заданной метки класса
        temp_acc = accuracy_score(
            temp_data_flt['t'].values,
            temp_data_flt['p'].values)
        # сохранение результата в словарь
        res[c] = temp_acc
    return res

def print_accuracy_score_for_classes(
    y_true: np.ndarray,
    y_pred: np.ndarray):
    """
    Вывод метрики accuracy для каждого класса
    """
    accs = accuracy_score_for_classes(y_true, y_pred)
    if len(accs)>0:
        print('Метка \t Accuracy')
    for i in accs:
        print('{} \t {}'.format(i, accs[i]))
```

In [79]:
```
vocabVect = CountVectorizer()
vocabVect.fit(data)
corpusVocab = vocabVect.vocabulary_
print('Количество сформированных признаков - {}'.format(len(corpusVocab)))
```

```
Количество сформированных признаков - 29970
```

In [80]:
```
for i in list(corpusVocab)[1:10]:
    print('{}={}'.format(i, corpusVocab[i]))
```

```
carl=7229
sol1=25350
gps=13433
caltech=7081
edu=10976
lydick=17538
subject=26157
re=22713
krillean=16582
```

Использование класса CountVectorizer

In [81]:
```
test_features = vocabVect.transform(data)
test_features
```

Out[81]:
```
<1785x29970 sparse matrix of type '<class 'numpy.int64'>'
	with 266767 stored elements in Compressed Sparse Row format>
```

In [82]:
```
test_features.todense()
```

Out[82]:
```
matrix([[0, 0, 0, ..., 0, 0, 0],
        [0, 0, 0, ..., 0, 0, 0],
        [0, 0, 0, ..., 0, 0, 0],
        ...,
        [1, 0, 0, ..., 0, 0, 0],
        [0, 0, 0, ..., 0, 0, 0],
        [0, 0, 0, ..., 0, 0, 0]])
```

In [83]:
```
# Размер нулевой строки
len(test_features.todense()[0].getA1())
```

Out[83]:
```
29970
```

In [84]:
```
# Непустые значения нулевой строки
print([i for i in test_features.todense()[0].getA1() if i>0])
```

```
[1, 1, 1, 1, 1, 1, 1, 1, 4, 3, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2, 4, 1, 1, 6, 1, 1,
1, 1, 1, 1, 1, 1, 1, 1, 1, 2, 3, 6, 1, 3, 1, 1, 1, 6, 1, 1, 1, 4, 1, 1, 1, 1, 1, 1, 1,
2, 1, 1, 1, 1, 2, 1, 1, 1, 1, 5, 7, 1, 3, 1, 3, 1, 1, 1, 1, 1, 2, 2, 2, 1, 4, 1, 1, 1,
4, 6, 1, 1, 4, 1, 3, 1, 1, 2, 1, 2, 1, 2, 1, 1, 1, 1, 1, 1, 2, 1, 2, 1, 1, 1, 1, 1, 1,
1, 5, 1, 2, 1, 1, 1, 1, 1, 6, 10, 1, 1, 1, 1, 8, 1, 1, 1, 1, 1, 1, 1, 2, 1, 1, 2, 1,
3, 1, 3, 1, 3, 1, 1, 1, 2, 1]
```

In [85]:
```
vocabVect.get_feature_names()[0:10]
```

```
/usr/local/lib/python3.7/dist-packages/sklearn/utils/deprecation.py:87: FutureWarning:
Function get_feature_names is deprecated; get_feature_names is deprecated in 1.0 and w
ill be removed in 1.2. Please use get_feature_names_out instead.
  warnings.warn(msg, category=FutureWarning)
```

Out[85]:
```
['00',
 '000',
 '0000',
 '0000001200',
 '00014',
 '000256',
 '001',
 '0010',
 '001004',
 '001323']
```

Решение задачи анализа тональности текста на основе модели "мешка слов"

In [86]:
```
def VectorizeAndClassify(vectorizers_list, classifiers_list):
    for v in vectorizers_list:
        for c in classifiers_list:
            pipeline1 = Pipeline([("vectorizer", v), ("classifier", c)])
            score = cross_val_score(pipeline1, newsgroups['data'], newsgroups['target
            print('Векторизация - {}'.format(v))
            print('Модель для классификации - {}'.format(c))
            print('Accuracy = {}'.format(score))
            print('===========================')
```

In [87]:
```
vectorizers_list = [CountVectorizer(vocabulary = corpusVocab)]
classifiers_list = [LogisticRegression(C=3.0), LinearSVC(), KNeighborsClassifier()]
VectorizeAndClassify(vectorizers_list, classifiers_list)
```

```
/usr/local/lib/python3.7/dist-packages/sklearn/linear_model/_logistic.py:818: Converge
nceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  extra_warning_msg=_LOGISTIC_SOLVER_CONVERGENCE_MSG,
/usr/local/lib/python3.7/dist-packages/sklearn/linear_model/_logistic.py:818: Converge
nceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  extra_warning_msg=_LOGISTIC_SOLVER_CONVERGENCE_MSG,
/usr/local/lib/python3.7/dist-packages/sklearn/linear_model/_logistic.py:818: Converge
nceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  extra_warning_msg=_LOGISTIC_SOLVER_CONVERGENCE_MSG,
Векторизация - CountVectorizer(vocabulary={'00': 0, '000': 1, '0000': 2, '0000001200':
3,
                            '00014': 4, '000256': 5, '001': 6, '0010': 7,
                            '001004': 8, '001323': 9, '001642': 10, '00309': 11,
                            '003221': 12, '003258u19250': 13, '0033': 14,
                            '003800': 15, '004021809': 16, '004158': 17,
                            '004418': 18, '004627': 19, '005': 20, '00500': 21,
                            '005148': 22, '005150': 23, '005512': 24,
                            '0059': 25, '007': 26, '0078': 27, '008': 28,
                            '008561': 29, ...})
Модель для классификации - LogisticRegression(C=3.0)
Accuracy = 0.9523809523809524
===========================
Векторизация - CountVectorizer(vocabulary={'00': 0, '000': 1, '0000': 2, '0000001200':
3,
                            '00014': 4, '000256': 5, '001': 6, '0010': 7,
                            '001004': 8, '001323': 9, '001642': 10, '00309': 11,
                            '003221': 12, '003258u19250': 13, '0033': 14,
                            '003800': 15, '004021809': 16, '004158': 17,
                            '004418': 18, '004627': 19, '005': 20, '00500': 21,
                            '005148': 22, '005150': 23, '005512': 24,
                            '0059': 25, '007': 26, '0078': 27, '008': 28,
                            '008561': 29, ...})
Модель для классификации - LinearSVC()
Accuracy = 0.9546218487394958
===========================
Векторизация - CountVectorizer(vocabulary={'00': 0, '000': 1, '0000': 2, '0000001200':
3,
                            '00014': 4, '000256': 5, '001': 6, '0010': 7,
                            '001004': 8, '001323': 9, '001642': 10, '00309': 11,
                            '003221': 12, '003258u19250': 13, '0033': 14,
                            '003800': 15, '004021809': 16, '004158': 17,
                            '004418': 18, '004627': 19, '005': 20, '00500': 21,
                            '005148': 22, '005150': 23, '005512': 24,
                            '0059': 25, '007': 26, '0078': 27, '008': 28,
                            '008561': 29, ...})
Модель для классификации - KNeighborsClassifier()
Accuracy = 0.687955182072829
===========================
```

Разделим выборку на обучающую и тестовую и проверим решение для лучшей модели

In [88]:
```
X_train, X_test, y_train, y_test = train_test_split(newsgroups['data'], newsgroups['ta
```

In [89]:
```
def sentiment(v, c):
    model = Pipeline(
        [("vectorizer", v),
         ("classifier", c)])
    model.fit(X_train, y_train)
    y_pred = model.predict(X_test)
    print_accuracy_score_for_classes(y_test, y_pred)
```

In [90]:
```
sentiment(CountVectorizer(), LinearSVC())
```

```
Метка    Accuracy
0        0.972972972972973
1        0.9084967320261438
2        0.9381443298969072
```

Работа с векторными представлениями слов с использованием word2vec

In [91]:
```
import gensim
from gensim.models import word2vec
```

In [101…
```
!wget https://rusvectores.org/static/models/rusvectores2/ruscorpora_mystem_cbow_300_2_

model_path = 'ruscorpora_mystem_cbow_300_2_2015.bin.gz'
```

```
--2022-06-01 01:55:35--  https://rusvectores.org/static/models/rusvectores2/ruscorpora
_mystem_cbow_300_2_2015.bin.gz
Resolving rusvectores.org (rusvectores.org)... 116.203.104.23
Connecting to rusvectores.org (rusvectores.org)|116.203.104.23|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 317128925 (302M) [application/x-gzip]
Saving to: ‘ruscorpora_mystem_cbow_300_2_2015.bin.gz’

ruscorpora_mystem_c 100%[===================>] 302.44M  14.5MB/s    in 22s

2022-06-01 01:55:58 (13.6 MB/s) - ‘ruscorpora_mystem_cbow_300_2_2015.bin.gz’ saved [31
7128925/317128925]
```

In [102…
```
model = gensim.models.KeyedVectors.load_word2vec_format(model_path, binary=True)
```

In [106…
```
words = ['праздник_S', 'елка_S', 'берег_S']
```

In [107…
```
for word in words:
    if word in model:
        print('\nСЛОВО - {}'.format(word))
        print('5 ближайших соседей слова:')
        for word, sim in model.most_similar(positive=[word], topn=5):
            print('{} => {}'.format(word, sim))
    else:
        print('Слово "{}" не найдено в модели'.format(word))
```

```
СЛОВО - праздник_S
5 ближайших соседей слова:
празднество_S => 0.6205120086669922
пасха_S => 0.6122827529907227
рождество_S => 0.573152482509613
новоселье_S => 0.5594838261604309
торжество_S => 0.5508145093917847

СЛОВО - елка_S
5 ближайших соседей слова:
елочка_S => 0.5322768092155457
березка_S => 0.46430379152297974
праздник_S => 0.44470053911209106
дерево_S => 0.4445313811302185
обсвечкать_V => 0.42405015230178833

СЛОВО - берег_S
5 ближайших соседей слова:
отмель_S => 0.6397713422775269
устье_S => 0.6352767944335938
излучина_S => 0.5934935808181763
река_S => 0.5909562110900879
побережье_S => 0.5815446376800537
```

Находим близость между словами и строим аналогии

In [110…

```
print(model.similarity('берег_S', 'озеро_S'))
```

```
0.5132777
```

In [111…

```
print(model.most_similar(positive=['праздник_S', 'пасха_S'], negative=['ханука_S']))
```

```
[('рождество_S', 0.5296328067779541), ('новоселье_S', 0.4783863425254822), ('воскресен
ие_S', 0.44232097268104553), ('сочельник_S', 0.439351886510849), ('именины_S', 0.43770
819902420044), ('свадьба_S', 0.42596834897994995), ('христосоваться_V', 0.410161018371
58203), ('благовещение_S', 0.4094565510749817), ('канун_S', 0.3997984230518341), ('пас
хальный_A', 0.39666759967803955)]
```

Обучим word2vec на наборе данных "fetch_20newsgroups"

In [112…

```
import re
import pandas as pd
import numpy as np
from typing import Dict, Tuple
from sklearn.metrics import accuracy_score, balanced_accuracy_score
from sklearn.feature_extraction.text import CountVectorizer, TfidfVectorizer
from sklearn.linear_model import LogisticRegression
from sklearn.pipeline import Pipeline
from nltk import WordPunctTokenizer
from nltk.corpus import stopwords
import nltk
nltk.download('stopwords')
```

```
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Unzipping corpora/stopwords.zip.
```

Out[112…

```
True
```

In [113…

```
categories = ["rec.motorcycles", "rec.sport.baseball", "sci.electronics","sci.med"]
newsgroups = fetch_20newsgroups(subset='train', categories=categories)
data = newsgroups['data']
```

In [114…

```
# Подготовим корпус
corpus = []
stop_words = stopwords.words('english')
tok = WordPunctTokenizer()
for line in newsgroups['data']:
    line1 = line.strip().lower()
    line1 = re.sub("[^a-zA-Z]"," ", line1)
    text_tok = tok.tokenize(line1)
    text_tok1 = [w for w in text_tok if not w in stop_words]
    corpus.append(text_tok1)
```

In [115…

```
corpus[:5]
```

Out[115…

```
[['nrmendel',
  'unix',
  'amherst',
  'edu',
  'nathaniel',
  'mendell',
```

```
 'subject',
 'bike',
 'advice',
 'organization',
 'amherst',
 'college',
 'x',
 'newsreader',
 'tin',
 'version',
 'pl',
 'lines',
 'ummm',
 'bikes',
 'kx',
 'suggest',
 'look',
 'zx',
 'since',
 'horsepower',
 'whereas',
 'might',
 'bit',
 'much',
 'sincerely',
 'nathaniel',
 'zx',
 'dod',
 'ama'],
['grante',
 'aquarius',
 'rosemount',
 'com',
 'grant',
 'edwards',
 'subject',
 'krillean',
 'photography',
 'reply',
 'grante',
 'aquarius',
 'rosemount',
 'com',
 'grant',
 'edwards',
 'organization',
 'rosemount',
 'inc',
 'lines',
 'nntp',
 'posting',
 'host',
 'aquarius',
 'stgprao',
 'st',
 'unocal',
 'com',
 'richard',
 'ottolini',
 'writes',
 'living',
 'things',
 'maintain',
 'small',
 'electric',
 'fields',
 'enhance',
 'certain',
 'chemical',
 'reactions',
 'promote',
 'communication',
 'states',
```

```
        'cell',
        'communicate',
        'cells',
        'nervous',
        'system',
        'specialized',
        'example',
        'perhaps',
        'uses',
        'true',
        'electric',
        'fields',
        'change',
        'location',
        'time',
        'large',
        'organism',
        'also',
        'true',
        'special',
        'photographic',
        'techniques',
        'applying',
        'external',
        'fields',
        'kirillian',
        'photography',
        'interact',
        'fields',
        'resistances',
        'caused',
        'fields',
        'make',
        'interesting',
        'pictures',
        'really',
        'kirlian',
        'photography',
        'taking',
        'pictures',
        'corona',
        'discharge',
        'objects',
        'animate',
        'inanimate',
        'fields',
        'applied',
        'objects',
        'millions',
        'times',
        'larger',
        'biologically',
        'created',
        'fields',
        'want',
        'record',
        'biologically',
        'created',
        'electric',
        'fields',
        'got',
        'use',
        'low',
        'noise',
        'high',
        'gain',
        'sensors',
        'typical',
        'eegs',
        'ekgs',
        'kirlian',
        'photography',
        'phun',
```

```
 'physics',
 'type',
 'stuff',
 'right',
 'soaking',
 'chunks',
 'extra',
 'fine',
 'steel',
 'wool',
 'liquid',
 'oxygen',
 'hitting',
 'hammer',
 'like',
 'kirlean',
 'setup',
 'fun',
 'possibly',
 'dangerous',
 'perhaps',
 'pictures',
 'diagonistic',
 'disease',
 'problems',
 'organisms',
 'better',
 'understood',
 'perhaps',
 'probably',
 'grant',
 'edwards',
 'yow',
 'vote',
 'rosemount',
 'inc',
 'well',
 'tapered',
 'half',
 'cocked',
 'ill',
 'conceived',
 'grante',
 'aquarius',
 'rosemount',
 'com',
 'tax',
 'deferred'],
['liny',
 'sun',
 'scri',
 'fsu',
 'edu',
 'nemo',
 'subject',
 'bates',
 'method',
 'myopia',
 'reply',
 'lin',
 'ray',
 'met',
 'fsu',
 'edu',
 'distribution',
 'na',
 'organization',
 'scri',
 'florida',
 'state',
 'university',
 'lines',
 'bates',
```

```
        'method',
        'work',
        'first',
        'heard',
        'newsgroup',
        'several',
        'years',
        'ago',
        'got',
        'hold',
        'book',
        'improve',
        'sight',
        'simple',
        'daily',
        'drills',
        'relaxation',
        'margaret',
        'corbett',
        'authorized',
        'instructor',
        'bates',
        'method',
        'published',
        'talks',
        'vision',
        'improvement',
        'relaxation',
        'exercise',
        'study',
        'whether',
        'method',
        'actually',
        'works',
        'works',
        'actually',
        'shortening',
        'previously',
        'elongated',
        'eyeball',
        'increasing',
        'lens',
        'ability',
        'flatten',
        'order',
        'compensate',
        'long',
        'eyeball',
        'since',
        'myopia',
        'result',
        'eyeball',
        'elongation',
        'seems',
        'logical',
        'approach',
        'correction',
        'find',
        'way',
        'reverse',
        'process',
        'e',
        'shorten',
        'somehow',
        'preferably',
        'non',
        'surgically',
        'recent',
        'studies',
        'find',
        'know',
        'rk',
        'works',
```

```
 'changing',
 'curvature',
 'cornea',
 'compensate',
 'shape',
 'eyeball',
 'way',
 'train',
 'muscles',
 'shorten',
 'eyeball',
 'back',
 'correct',
 'length',
 'would',
 'even',
 'better',
 'bates',
 'idea',
 'right',
 'thanks',
 'information'],
['mcovingt',
 'aisun',
 'ai',
 'uga',
 'edu',
 'michael',
 'covington',
 'subject',
 'buy',
 'parts',
 'time',
 'nntp',
 'posting',
 'host',
 'aisun',
 'ai',
 'uga',
 'edu',
 'organization',
 'ai',
 'programs',
 'university',
 'georgia',
 'athens',
 'lines',
 'pricing',
 'parts',
 'reminds',
 'something',
 'chemist',
 'said',
 'gram',
 'dye',
 'costs',
 'dollar',
 'comes',
 'liter',
 'jar',
 'also',
 'costs',
 'dollar',
 'want',
 'whole',
 'barrel',
 'also',
 'costs',
 'dollar',
 'e',
 'charge',
 'almost',
```

```
 'exclusively',
 'packaging',
 'delivering',
 'chemical',
 'particular',
 'case',
 'byproduct',
 'cost',
 'almost',
 'nothing',
 'intrinsically',
 'michael',
 'covington',
 'associate',
 'research',
 'scientist',
 'artificial',
 'intelligence',
 'programs',
 'mcovingt',
 'ai',
 'uga',
 'edu',
 'university',
 'georgia',
 'phone',
 'athens',
 'georgia',
 'u',
 'amateur',
 'radio',
 'n',
 'tmi'],
['tammy',
 'vandenboom',
 'launchpad',
 'unc',
 'edu',
 'tammy',
 'vandenboom',
 'subject',
 'sore',
 'spot',
 'testicles',
 'nntp',
 'posting',
 'host',
 'lambada',
 'oit',
 'unc',
 'edu',
 'organization',
 'university',
 'north',
 'carolina',
 'extended',
 'bulletin',
 'board',
 'service',
 'distribution',
 'na',
 'lines',
 'husband',
 'woke',
 'three',
 'days',
 'ago',
 'small',
 'sore',
 'spot',
 'spot',
 'size',
 'nickel',
```

```
 'one',
 'testicles',
 'bottom',
 'side',
 'knots',
 'lumps',
 'little',
 'sore',
 'spot',
 'says',
 'reminds',
 'bruise',
 'feels',
 'recollection',
 'hitting',
 'anything',
 'like',
 'would',
 'cause',
 'bruise',
 'asssures',
 'remember',
 'something',
 'like',
 'clues',
 'might',
 'somewhat',
 'hypochondriac',
 'sp',
 'sure',
 'gonna',
 'die',
 'thanks',
 'opinions',
 'expressed',
 'necessarily',
 'university',
 'north',
 'carolina',
 'chapel',
 'hill',
 'campus',
 'office',
 'information',
 'technology',
 'experimental',
 'bulletin',
 'board',
 'service',
 'internet',
 'launchpad',
 'unc',
 'edu']]
```

In [116…

```
%time model_imdb = word2vec.Word2Vec(corpus, workers=4, min_count=10, window=10, sampl
```

```
CPU times: user 4.73 s, sys: 23.8 ms, total: 4.75 s
Wall time: 2.95 s
```

In [117…

```
# Проверим, что модель обучилась
print(model_imdb.wv.most_similar(positive=['find'], topn=5))
```

```
[('using', 0.9903143644332886), ('circuits', 0.9873676300048828), ('used', 0.986776113
5101318), ('work', 0.9865034818649292), ('etc', 0.9864989519119263)]
```

In [118…

```
def sentiment_2(v, c):
    model = Pipeline(
        [("vectorizer", v),
         ("classifier", c)])
    model.fit(X_train, y_train)
```

```
    y_pred = model.predict(X_test)
    print_accuracy_score_for_classes(y_test, y_pred)
```

Проверка качества работы модели word2vec

In [119…

```
class EmbeddingVectorizer(object):
    '''
    Для текста усредним вектора входящих в него слов
    '''
    def __init__(self, model):
        self.model = model
        self.size = model.vector_size

    def fit(self, X, y):
        return self

    def transform(self, X):
        return np.array([np.mean(
            [self.model[w] for w in words if w in self.model]
            or [np.zeros(self.size)], axis=0)
            for words in X])
```

In [120…

```
def accuracy_score_for_classes(
    y_true: np.ndarray,
    y_pred: np.ndarray) -> Dict[int, float]:
    """
    Вычисление метрики accuracy для каждого класса
    y_true - истинные значения классов
    y_pred - предсказанные значения классов
    Возвращает словарь: ключ - метка класса,
    значение - Accuracy для данного класса
    """
    # Для удобства фильтрации сформируем Pandas DataFrame
    d = {'t': y_true, 'p': y_pred}
    df = pd.DataFrame(data=d)
    # Метки классов
    classes = np.unique(y_true)
    # Результирующий словарь
    res = dict()
    # Перебор меток классов
    for c in classes:
        # отфильтруем данные, которые соответствуют
        # текущей метке класса в истинных значениях
        temp_data_flt = df[df['t']==c]
        # расчет accuracy для заданной метки класса
        temp_acc = accuracy_score(
            temp_data_flt['t'].values,
            temp_data_flt['p'].values)
        # сохранение результата в словарь
        res[c] = temp_acc
    return res

def print_accuracy_score_for_classes(
    y_true: np.ndarray,
    y_pred: np.ndarray):
    """
    Вывод метрики accuracy для каждого класса
    """
    accs = accuracy_score_for_classes(y_true, y_pred)
    if len(accs)>0:
        print('Метка \t Accuracy')
    for i in accs:
        print('{} \t {}'.format(i, accs[i]))
```

In [121…

```
# Обучающая и тестовая выборки
boundary = 1500
X_train = corpus[:boundary]
X_test = corpus[boundary:]
y_train = newsgroups['target'][:boundary]
y_test = newsgroups['target'][boundary:]
```

In [122…

```
sentiment_2(EmbeddingVectorizer(model_imdb.wv), LogisticRegression(C=5.0))
```

```
/usr/local/lib/python3.7/dist-packages/sklearn/linear_model/_logistic.py:818: Converge
nceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  extra_warning_msg=_LOGISTIC_SOLVER_CONVERGENCE_MSG,
```

Метка	Accuracy
0	0.8333333333333334
1	0.9223300970873787
2	0.7247706422018348
3	0.7192982456140351

Как видно из результатов проверки качества моделей, лучшее качество показал CountVectorizer