Лабораторная работа по теме "Классификация текста"

Выполнил: Пакало Александр Сергеевич ИУ5-22М

Задание

Для произвольного набора данных, предназначенного для классификации текстов, решите задачу классификации текста двумя способами:

- Способ 1. На основе CountVectorizer или **TfidfVectorizer**.
- Способ 2. На основе моделей **word2vec** или Glove или fastText.

Сравните качество полученных моделей. Для поиска наборов данных в поисковой системе можно использовать ключевые слова "datasets for text classification".

In [58]:
```
# Loading extension for reloading editable packages (pip install -e .)
%load_ext autoreload
```

In [1]:
```
RANDOM_SEED = 13
```

Набор данных

Проведём классификацию текста используя набор данных [BBC News Archive (https://www.kaggle.com/datasets/hgultekin/bbcnewsarchive)](https://www.kaggle.com/datasets/hgultekin/bbcnewsarchive)

Подготовка переменных для работы с данными

In [2]:
```
from pathlib import Path

data_path = Path("../../data")
external_data_path = data_path / "external"
raw_data_path = data_path / "raw"

dataset_filename = "bbc-news-data.zip"
```

Разархивирование набора данных

In [3]:
```
import os
import shutil

raw_data_path.mkdir(exist_ok=True)

file_path = external_data_path / dataset_filename
raw_data_path = external_data_path / dataset_filename

if not os.path.isfile(raw_data_path):
    shutil.unpack_archive(file_path, extract_dir=raw_data_path)
    # file_path.unlink()  # Remove archive after extracting it.
```

Загрузка данных из csv

In [4]:
```
import pandas as pd

df = pd.read_csv(raw_data_path, sep="\t")
```

Разведочный анализ данных

Ознакомимся немного с данными, с которыми собираемся работать

Основные характеристики датасета

In [5]:
```
df.head()
```

Out[5]:

	category	filename	title	content
0	business	001.txt	Ad sales boost Time Warner profit	Quarterly profits at US media giant TimeWarne...
1	business	002.txt	Dollar gains on Greenspan speech	The dollar has hit its highest level against ...
2	business	003.txt	Yukos unit buyer faces loan claim	The owners of embattled Russian oil giant Yuk...
3	business	004.txt	High fuel prices hit BA's profits	British Airways has blamed high fuel prices f...
4	business	005.txt	Pernod takeover talk lifts Domecq	Shares in UK drinks and food firm Allied Dome...

In [6]:
```
df.tail()
```

Out[6]:

	category	filename	title	content
2220	tech	397.txt	BT program to beat dialler scams	BT is introducing two initiatives to help bea...
2221	tech	398.txt	Spam e-mails tempt net shoppers	Computer users across the world continue to i...
2222	tech	399.txt	Be careful how you code	A new European directive could put software w...
2223	tech	400.txt	US cyber security chief resigns	The man making sure US computer networks are ...
2224	tech	401.txt	Losing yourself in online gaming	Online role playing games are time-consuming,...

Размер датасета

In [7]:
```
num_of_rows, num_of_columns = df.shape
print(f'Размер датасета: {num_of_rows} строк, {num_of_columns} колонок'
```

```
Размер датасета: 2225 строк, 4 колонок
```

Определение типов

In [8]:
```
df.dtypes
```

Out[8]:
```
category    object
filename    object
title       object
content     object
dtype: object
```

Проверка на наличие пустых значений

In [9]:
```
df.isnull().sum()
```

Out[9]:
```
category    0
filename    0
title       0
content     0
dtype: int64
```

Обработки пустых значений не требуется

Проверка на уникальные значения

In [10]:
```
pd.Series(df["category"].unique())
```

Out[10]:
```
0         business
1    entertainment
2         politics
3            sport
4             tech
dtype: object
```

Подготовка корпуса

Некоторые колонки имеют неверные типы данных, их следует преобразовать.

Строки вместо `object` сделаем `string`, а колонку "category" сделаем типа `category`.

In [11]:
```
df = df.astype({
    "category": "category",
    "filename": "string",
    "title": "string",
    "content": "string",
})
df.dtypes
```

Out[11]:
```
category          category
filename    string[python]
title       string[python]
content     string[python]
dtype: object
```

Токенизация

Загрузка модели spacy.

In [12]:
```
import spacy

spacy_prefers_gpu = spacy.prefer_gpu()
nlp = spacy.load("en_core_web_sm")
```

Токенизация текстовых значений набора данных (кроме названия файла)

In [13]:
```python
from spacy.tokens.doc import Doc

corpus: list[Doc] = []

for text in (df["title"] + df["content"]).values:
    corpus.append(nlp(text))

corpus[:3]
```

Out[13]:
```
[Ad sales boost Time Warner profit Quarterly profits at US media gia
nt TimeWarner jumped 76% to $1.13bn (£600m) for the three months to
December, from $639m year-earlier.  The firm, which is now one of th
e biggest investors in Google, benefited from sales of high-speed in
ternet connections and higher advert sales. TimeWarner said fourth q
uarter sales rose 2% to $11.1bn from $10.9bn. Its profits were buoye
d by one-off gains which offset a profit dip at Warner Bros, and les
s users for AOL.  Time Warner said on Friday that it now owns 8% of
search-engine Google. But its own internet business, AOL, had has mi
xed fortunes. It lost 464,000 subscribers in the fourth quarter prof
its were lower than in the preceding three quarters. However, the co
mpany said AOL's underlying profit before exceptional items rose 8%
on the back of stronger internet advertising revenues. It hopes to i
ncrease subscribers by offering the online service free to TimeWarne
r internet customers and will try to sign up AOL's existing customer
s for high-speed broadband. TimeWarner also has to restate 2000 and
2003 results following a probe by the US Securities Exchange Commiss
ion (SEC), which is close to concluding.  Time Warner's fourth quart
er profits were slightly better than analysts' expectations. But its
film division saw profits slump 27% to $284m, helped by box-office f
lops Alexander and Catwoman, a sharp contrast to year-earlier, when
the third and final film in the Lord of the Rings trilogy boosted re
sults. For the full-year, TimeWarner posted a profit of $3.36bn, up
27% from its 2003 performance, while revenues grew 6.4% to $42.09bn.
"Our financial performance was strong, meeting or exceeding all of o
ur full-year objectives and greatly enhancing our flexibility," chai
rman and chief executive Richard Parsons said. For 2005, TimeWarner
is projecting operating earnings growth of around 5%, and also expec
ts higher revenue and wider profit margins.  TimeWarner is to restat
e its accounts as part of efforts to resolve an inquiry into AOL by
US market regulators. It has already offered to pay $300m to settle
charges, in a deal that is under review by the SEC. The company said
it was unable to estimate the amount it needed to set aside for lega
l reserves, which it previously set at $500m. It intends to adjust t
he way it accounts for a deal with German music publisher Bertelsman
n's purchase of a stake in AOL Europe, which it had reported as adve
rtising revenue. It will now book the sale of its stake in AOL Europ
e as a loss on the value of that stake. ,
 Dollar gains on Greenspan speech The dollar has hit its highest lev
el against the euro in almost three months after the Federal Reserve
head said the US trade deficit is set to stabilise.  And Alan Greens
pan highlighted the US government's willingness to curb spending and
rising household savings as factors which may help to reduce it. In
late trading in New York, the dollar reached $1.2871 against the eur
o, from $1.2974 on Thursday. Market concerns about the deficit has h
it the greenback in recent months. On Friday, Federal Reserve chairm
an Mr Greenspan's speech in London ahead of the meeting of G7 financ
e ministers sent the dollar higher after it had earlier tumbled on t
he back of worse-than-expected US jobs data. "I think the chairman's
taking a much more sanguine view on the current account deficit than
```

```
he's taken for some time," said Robert Sinche, head of currency stra
tegy at Bank of America in New York. "He's taking a longer-term view
, laying out a set of conditions under which the current account def
icit can improve this year and next."  Worries about the deficit con
cerns about China do, however, remain. China's currency remains pegg
ed to the dollar and the US currency's sharp falls in recent months
have therefore made Chinese export prices highly competitive. But ca
lls for a shift in Beijing's policy have fallen on deaf ears, despit
e recent comments in a major Chinese newspaper that the "time is rip
e" for a loosening of the peg. The G7 meeting is thought unlikely to
produce any meaningful movement in Chinese policy. In the meantime,
the US Federal Reserve's decision on 2 February to boost interest ra
tes by a quarter of a point - the sixth such move in as many months
- has opened up a differential with European rates. The half-point w
indow, some believe, could be enough to keep US assets looking more
attractive, and could help prop up the dollar. The recent falls have
partly been the result of big budget deficits, as well as the US's y
awning current account gap, both of which need to be funded by the b
uying of US bonds and assets by foreign firms and governments. The W
hite House will announce its budget on Monday, and many commentators
believe the deficit will remain at close to half a trillion dollars.
,
 Yukos unit buyer faces loan claim The owners of embattled Russian o
il giant Yukos are to ask the buyer of its former production unit to
pay back a $900m (£479m) loan.  State-owned Rosneft bought the Yugan
sk unit for $9.3bn in a sale forced by Russia to part settle a $27.5
bn tax claim against Yukos. Yukos' owner Menatep Group says it will
ask Rosneft to repay a loan that Yugansk had secured on its assets.
Rosneft already faces a similar $540m repayment demand from foreign
banks. Legal experts said Rosneft's purchase of Yugansk would includ
e such obligations. "The pledged assets are with Rosneft, so it will
have to pay real money to the creditors to avoid seizure of Yugansk
assets," said Moscow-based US lawyer Jamie Firestone, who is not con
nected to the case. Menatep Group's managing director Tim Osborne to
ld the Reuters news agency: "If they default, we will fight them whe
re the rule of law exists under the international arbitration clause
s of the credit."  Rosneft officials were unavailable for comment. B
ut the company has said it intends to take action against Menatep to
recover some of the tax claims and debts owed by Yugansk. Yukos had
filed for bankruptcy protection in a US court in an attempt to preve
nt the forced sale of its main production arm. The sale went ahead i
n December and Yugansk was sold to a little-known shell company whic
h in turn was bought by Rosneft. Yukos claims its downfall was punis
hment for the political ambitions of its founder Mikhail Khodorkovsk
y and has vowed to sue any participant in the sale. ]
```

In [14]:
```python
assert len(corpus) == num_of_rows
```

Заметим, что spacy при печати и выводит текст, на самом деле это объект. Word2Vec ожидает увидеть строки либо списки.

In [15]:
```python
spacy_text = nlp('training: nlp!')
spacy_text, type(spacy_text), type(spacy_text[0])
```

Out[15]:
```
(training: nlp!, spacy.tokens.doc.Doc, spacy.tokens.token.Token)
```

In [16]:
```
[token.text for token in spacy_text]
```

Out[16]:
```
['training', ':', 'nlp', '!']
```

Поэтому преобразуем наш `corpus` в упрощённый формат, совместимый с word2vec.

In [17]:
```
corpus_for_word2vec = [[token.text for token in spacy_text] for spacy_t

corpus_for_word2vec[:3]
```

Out[17]:
```
[['Ad',
  'sales',
  'boost',
  'Time',
  'Warner',
  'profit',
  'Quarterly',
  'profits',
  'at',
  'US',
  'media',
  'giant',
  'TimeWarner',
  'jumped',
  '76',
  '%',
  'to',
  '$',
  '1.13bn',
```

Модель word2vec

Список доступных предобученнных моделей

In [18]:
```
from gensim.downloader import info

list(info()['models'].keys())
```

Out[18]:
```
['fasttext-wiki-news-subwords-300',
 'conceptnet-numberbatch-17-06-300',
 'word2vec-ruscorpora-300',
 'word2vec-google-news-300',
 'glove-wiki-gigaword-50',
 'glove-wiki-gigaword-100',
 'glove-wiki-gigaword-200',
 'glove-wiki-gigaword-300',
 'glove-twitter-25',
 'glove-twitter-50',
 'glove-twitter-100',
 'glove-twitter-200',
 '__testing_word2vec-matrix-synopsis']
```

Загрузка модели

In [19]:
```python
from gensim.downloader import info, load

# You can safely restart this cell, gensim will download it only once.
# It still takes some time to load, though.
word2vec_google_news_300_model = load("word2vec-google-news-300")
```

Небольшая проверка работы модели

In [20]:
```python
words = ["king", "queen", "man", "woman"]
```

In [21]:
```python
from itertools import permutations

PAIRS = 2
[f"Для пары слов {word_pair} схожесть: {word2vec_google_news_300_model.
```

Out[21]:
```
["Для пары слов ('king', 'queen') схожесть: 0.65",
 "Для пары слов ('king', 'man') схожесть: 0.23",
 "Для пары слов ('king', 'woman') схожесть: 0.13",
 "Для пары слов ('queen', 'king') схожесть: 0.65",
 "Для пары слов ('queen', 'man') схожесть: 0.17",
 "Для пары слов ('queen', 'woman') схожесть: 0.32",
 "Для пары слов ('man', 'king') схожесть: 0.23",
 "Для пары слов ('man', 'queen') схожесть: 0.17",
 "Для пары слов ('man', 'woman') схожесть: 0.77",
 "Для пары слов ('woman', 'king') схожесть: 0.13",
 "Для пары слов ('woman', 'queen') схожесть: 0.32",
 "Для пары слов ('woman', 'man') схожесть: 0.77"]
```

Обучение собственной модели

In [22]:
```
corpus[0].text
```

Out[22]:
```
'Ad sales boost Time Warner profit Quarterly profits at US media gia
nt TimeWarner jumped 76% to $1.13bn (£600m) for the three months to
December, from $639m year-earlier.  The firm, which is now one of th
e biggest investors in Google, benefited from sales of high-speed in
ternet connections and higher advert sales. TimeWarner said fourth q
uarter sales rose 2% to $11.1bn from $10.9bn. Its profits were buoye
d by one-off gains which offset a profit dip at Warner Bros, and les
s users for AOL.  Time Warner said on Friday that it now owns 8% of
search-engine Google. But its own internet business, AOL, had has mi
xed fortunes. It lost 464,000 subscribers in the fourth quarter prof
its were lower than in the preceding three quarters. However, the co
mpany said AOL\'s underlying profit before exceptional items rose 8%
on the back of stronger internet advertising revenues. It hopes to i
ncrease subscribers by offering the online service free to TimeWarne
r internet customers and will try to sign up AOL\'s existing custome
rs for high-speed broadband. TimeWarner also has to restate 2000 and
2003 results following a probe by the US Securities Exchange Commiss
ion (SEC), which is close to concluding.  Time Warner\'s fourth quar
ter profits were slightly better than analysts\' expectations. But i
ts film division saw profits slump 27% to $284m, helped by box-offic
e flops Alexander and Catwoman, a sharp contrast to year-earlier, wh
en the third and final film in the Lord of the Rings trilogy boosted
results. For the full-year, TimeWarner posted a profit of $3.36bn, u
p 27% from its 2003 performance, while revenues grew 6.4% to $42.09b
n. "Our financial performance was strong, meeting or exceeding all o
f our full-year objectives and greatly enhancing our flexibility," c
hairman and chief executive Richard Parsons said. For 2005, TimeWarn
er is projecting operating earnings growth of around 5%, and also ex
pects higher revenue and wider profit margins.  TimeWarner is to res
tate its accounts as part of efforts to resolve an inquiry into AOL
by US market regulators. It has already offered to pay $300m to sett
le charges, in a deal that is under review by the SEC. The company s
aid it was unable to estimate the amount it needed to set aside for
legal reserves, which it previously set at $500m. It intends to adju
st the way it accounts for a deal with German music publisher Bertel
smann\'s purchase of a stake in AOL Europe, which it had reported as
advertising revenue. It will now book the sale of its stake in AOL E
urope as a loss on the value of that stake. '
```

In [23]:
```
from gensim.models import word2vec

model_trained_on_dataset: word2vec.Word2Vec

%time model_trained_on_dataset = word2vec.Word2Vec(corpus_for_word2vec,
```

```
CPU times: user 3.55 s, sys: 5.56 ms, total: 3.56 s
Wall time: 764 ms
```

In [24]:
```
wv = model_trained_on_dataset.wv
```

In [25]:
```python
for index, word in enumerate(wv.index_to_key):
    if index == 10:
        break
    print(f"word #{index}/{len(wv.index_to_key)} is {word}")
```

```
word #0/7528 is the
word #1/7528 is .
word #2/7528 is ,
word #3/7528 is to
word #4/7528 is "
word #5/7528 is of
word #6/7528 is and
word #7/7528 is a
word #8/7528 is in
word #9/7528 is -
```

In [26]:
```
wv['the']
```

Out[26]:
```
array([ 2.2328116e-03,  1.3145196e+00,  4.8119223e-01,  8.2273012e-0
1,
        -7.3784679e-01,  1.1799921e-01,  1.2944862e-01, -1.8478009e-0
1,
        -9.5712818e-02,  4.8478240e-01,  1.3814596e+00,  6.6620255e-0
1,
        -4.7658932e-01,  3.2528889e-01, -7.1216339e-01,  5.6120133e-0
1,
         8.8995785e-01, -4.8151794e-01, -1.4412236e+00, -1.8842486e+0
0,
         1.0169474e+00,  8.3980924e-01,  2.4093542e+00, -9.2637980e-0
1,
        -8.3508015e-01,  9.9907434e-01, -5.7357568e-01, -1.4646709e+0
0,
        -6.7030293e-01, -3.8889077e-01, -7.6762259e-01,  2.7496248e-0
1,
         1.0931668e+00,  6.8747324e-01,  6.2028271e-01, -3.3290204e-0
1,
         5.7881808e-01,  5.3475720e-01,  1.8358558e-01,  2.6981041e-0
1,
        -2.4757305e-01,  2.0826844e-02, -4.2661145e-01,  3.0817699e-0
1,
        -2.9780325e-01,  9.1499192e-01, -1.1656585e+00,  1.4410807e+0
0,
         1.2741096e+00, -6.7983754e-02, -1.7749549e+00, -1.9587585e-0
1,
         2.4553971e-01, -7.5158238e-01,  5.5168909e-01, -1.2906446e-0
1,
         4.2212862e-01,  7.7437592e-01,  5.7517707e-01,  6.7730933e-0
1,
         8.8157815e-01, -7.6260799e-01, -4.7266704e-01,  2.6451347e-0
2,
        -7.6551944e-02,  2.3745088e-02,  1.9332466e-01,  5.6181389e-0
1,
         4.4821772e-01,  7.7733681e-02,  3.2478690e-01,  3.0262700e-0
1,
        -6.9305700e-01, -9.7270983e-01, -5.7941396e-04,  1.0702924e+0
0,
        -2.6772329e-01,  3.6738709e-01,  6.1123437e-01,  1.0923735e+0
0,
        -3.4841591e-01,  5.8158273e-01,  2.2036607e+00,  3.7529814e-0
1,
        -4.2797166e-01, -4.8759389e-01,  7.7847892e-01, -1.3478311e+0
0,
         7.5376707e-01, -1.8138057e-01,  4.5300011e-02,  9.0197438e-0
1,
         8.1529826e-01, -2.1493059e-01,  7.5356930e-01, -1.8846135e-0
1,
        -2.4992822e-01, -3.3751857e-01, -4.7414747e-01,  8.6240608e-0
1],
      dtype=float32)
```

In [27]:
```
model_trained_on_dataset.wv.most_similar(positive=['the'], topn=5)
```

Out[27]:
```
[('tackled', 0.5361359715461731),
 ('question', 0.5353929400444031),
 ('House', 0.5064075589179993),
 ('another', 0.49653181433677673),
 ('course', 0.4722856879234314)]
```

Классификация

Подготовка данных для классификации

Выберем `X` и `y` среди нашего набора данных

In [28]:
```
X = corpus_for_word2vec
y = df["category"].values
```

Составим выборки для обучения

In [29]:
```
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(
    X, y,
    test_size=0.33,
    random_state=RANDOM_SEED
)
```

Аналогичные выборки сделаем для моделей, которые в себя принимают список строк, а не список токенов.

In [30]:
```
X_str = [spacy_text.text for spacy_text in corpus]
```

In [31]:
```
from sklearn.model_selection import train_test_split

X_str_train, X_str_test, y_str_train, y_str_test = train_test_split(
    X_str, y,
    test_size=0.33,
    random_state=RANDOM_SEED
)
```

Составим pipeline

Подготовим scaler'ы для последующих моделей.

Например, NaiveBayes умеет работать только с неотрицательными числами.

In [32]:
```python
from sklearn.preprocessing import MinMaxScaler, StandardScaler

standard_scaler = StandardScaler()
min_max_scaler = MinMaxScaler()
```

Составим pipeline

In [33]:
```python
import numpy as np
from IPython.display import display
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, balanced_accuracy_score


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)
    results = pd.DataFrame(data={ "Категория": accs.keys(), "Точность":

    display(results)

    return results
```

In [34]:
```
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 [35]:
```
from sklearn.pipeline import Pipeline


def classifier_pipeline(v, c, scaler=None, corpus_already_tokenized=Tru
    pipeline_steps = [
        ("vectorizer", v),
    ]

    if scaler:
        pipeline_steps.append(("scaler", scaler))

    pipeline_steps.append(("classifier", c))

    pipeline = Pipeline(pipeline_steps)

    classifier_X_train = X_train
    classifier_y_train = y_train
    classifier_X_test = X_test
    classifier_y_test = y_test

    if not corpus_already_tokenized:
        classifier_X_train = X_str_train
        classifier_y_train = y_str_train
        classifier_X_test = X_str_test
        classifier_y_test = y_str_test

    pipeline.fit(classifier_X_train, classifier_y_train)
    y_pred = pipeline.predict(classifier_X_test)

    return print_accuracy_score_for_classes(classifier_y_test, y_pred)
```

Проверка результатов

In [36]:
```
ClassifierName = str
ModelName = str

metrics_data: dict[ClassifierName, dict[ModelName, float]] = {}
```

In [37]:
```
def add_metrics_data(classier_name: ClassifierName, model_name: ModelNa
    if not metrics_data.get(classier_name):
        metrics_data[classier_name] = {}

    metrics_data[classier_name][model_name] = np.mean(results['Точность

    return metrics_data
```

Протестируем собственно-обученную модель Word2Vec.

LogisticRegression

In [38]:
```
add_metrics_data("LogisticRegression", "our w2v", classifier_pipeline(E
```

```
/home/ds13/Projects/--educational/Bmstu__/t5-2-/MachineLearningMetho
ds__/t5-2-MachineLearningMethods__Fundamentals/.venv/lib/python3.12/
site-packages/sklearn/linear_model/_logistic.py:469: ConvergenceWarn
ing: 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 sh
own in:
    https://scikit-learn.org/stable/modules/preprocessing.html (http
s://scikit-learn.org/stable/modules/preprocessing.html)
Please also refer to the documentation for alternative solver option
s:
    https://scikit-learn.org/stable/modules/linear_model.html#logist
ic-regression (https://scikit-learn.org/stable/modules/linear_mode
l.html#logistic-regression)
  n_iter_i = _check_optimize_result(
```

	Категория	Точность
0	business	0.861635
1	entertainment	0.766423
2	politics	0.870504
3	sport	0.924528
4	tech	0.879433

Out[38]:
```
{'LogisticRegression': {'our w2v': 0.8605046201825178}}
```

LogisticRegression c min_max_scaler

In [39]:
```
add_metrics_data("LogisticRegression with scaler", "our w2v", classifie
```

```
/home/ds13/Projects/--educational/Bmstu__/t5-2-/MachineLearningMetho
ds__/t5-2-MachineLearningMethods__Fundamentals/.venv/lib/python3.12/
site-packages/sklearn/linear_model/_logistic.py:469: ConvergenceWarn
ing: 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 sh
own in:
    https://scikit-learn.org/stable/modules/preprocessing.html (http
s://scikit-learn.org/stable/modules/preprocessing.html)
Please also refer to the documentation for alternative solver option
s:
    https://scikit-learn.org/stable/modules/linear_model.html#logist
ic-regression (https://scikit-learn.org/stable/modules/linear_mode
l.html#logistic-regression)
  n_iter_i = _check_optimize_result(
```

	Категория	Точность
0	business	0.861635
1	entertainment	0.781022
2	politics	0.884892
3	sport	0.924528
4	tech	0.865248

Out[39]:
```
{'LogisticRegression': {'our w2v': 0.8605046201825178},
 'LogisticRegression with scaler': {'our w2v': 0.8634651466208176}}
```

MultinomialNaiveBayes

In [40]:
```
from sklearn.naive_bayes import MultinomialNB

# NB нужны неотрицательные значения.
add_metrics_data("MultinomialNB", "our w2v", classifier_pipeline(Embedd
```

	Категория	Точность
0	business	0.792453
1	entertainment	0.554745
2	politics	0.791367
3	sport	0.779874
4	tech	0.801418

Out[40]:
```
{'LogisticRegression': {'our w2v': 0.8605046201825178},
 'LogisticRegression with scaler': {'our w2v': 0.8634651466208176},
 'MultinomialNB': {'our w2v': 0.743971383152747}}
```

KNeighborsClassifier

In [41]:
```python
from sklearn.neighbors import KNeighborsClassifier

# KNC нужны значения, распределённые по нормальному распределению.
add_metrics_data("KNeighborsClassifier", "our w2v", classifier_pipeline
```

	Категория	Точность
0	business	0.836478
1	entertainment	0.708029
2	politics	0.856115
3	sport	0.893082
4	tech	0.858156

Out[41]:
```
{'LogisticRegression': {'our w2v': 0.8605046201825178},
 'LogisticRegression with scaler': {'our w2v': 0.8634651466208176},
 'MultinomialNB': {'our w2v': 0.743971383152747},
 'KNeighborsClassifier': {'our w2v': 0.8303720163580858}}
```

DecisionTreeClassifier

In [42]:
```python
from sklearn.tree import DecisionTreeClassifier

classifier_pipeline(EmbeddingVectorizer(model_trained_on_dataset.wv), D
```

	Категория	Точность
0	business	0.735849
1	entertainment	0.635036
2	politics	0.762590
3	sport	0.792453
4	tech	0.822695

Out[42]:

	Категория	Точность
0	business	0.735849
1	entertainment	0.635036
2	politics	0.762590
3	sport	0.792453
4	tech	0.822695

In [43]:
```
classifier_pipeline(EmbeddingVectorizer(model_trained_on_dataset.wv), D
```

	Категория	Точность
0	business	0.729560
1	entertainment	0.598540
2	politics	0.741007
3	sport	0.817610
4	tech	0.787234

Out[43]:

	Категория	Точность
0	business	0.729560
1	entertainment	0.598540
2	politics	0.741007
3	sport	0.817610
4	tech	0.787234

In [44]:
```
add_metrics_data("DecisionTreeClassifier", "our w2v", classifier_pipeli
```

	Категория	Точность
0	business	0.779874
1	entertainment	0.620438
2	politics	0.733813
3	sport	0.805031
4	tech	0.829787

Out[44]:
```
{'LogisticRegression': {'our w2v': 0.8605046201825178},
 'LogisticRegression with scaler': {'our w2v': 0.8634651466208176},
 'MultinomialNB': {'our w2v': 0.743971383152747},
 'KNeighborsClassifier': {'our w2v': 0.8303720163580858},
 'DecisionTreeClassifier': {'our w2v': 0.7537887600529158}}
```

Протестируем предобученную модель от google Word2Vec.

LogisticRegression

In [45]:
```
add_metrics_data("LogisticRegression", "google w2v", classifier_pipelin
```

	Категория	Точность
0	business	0.943396
1	entertainment	0.963504
2	politics	0.949640
3	sport	0.993711
4	tech	0.971631

Out[45]:
```
{'LogisticRegression': {'our w2v': 0.8605046201825178,
  'google w2v': 0.9643764122635148},
 'LogisticRegression with scaler': {'our w2v': 0.8634651466208176},
 'MultinomialNB': {'our w2v': 0.743971383152747},
 'KNeighborsClassifier': {'our w2v': 0.8303720163580858},
 'DecisionTreeClassifier': {'our w2v': 0.7537887600529158}}
```

LogisticRegression с min_max_scaler

In [46]:
```
add_metrics_data("LogisticRegression with scaler", "google w2v", classi
```

	Категория	Точность
0	business	0.955975
1	entertainment	0.978102
2	politics	0.942446
3	sport	0.993711
4	tech	0.992908

Out[46]:
```
{'LogisticRegression': {'our w2v': 0.8605046201825178,
  'google w2v': 0.9643764122635148},
 'LogisticRegression with scaler': {'our w2v': 0.8634651466208176,
  'google w2v': 0.9726283137912249},
 'MultinomialNB': {'our w2v': 0.743971383152747},
 'KNeighborsClassifier': {'our w2v': 0.8303720163580858},
 'DecisionTreeClassifier': {'our w2v': 0.7537887600529158}}
```

MultinomialNaiveBayes

In [47]:
```
from sklearn.naive_bayes import MultinomialNB

# NB нужны неотрицательные значения.
add_metrics_data("MultinomialNB", "google w2v", classifier_pipeline(Emb
```

	Категория	Точность
0	business	0.924528
1	entertainment	0.905109
2	politics	0.949640
3	sport	0.987421
4	tech	0.929078

Out[47]:
```
{'LogisticRegression': {'our w2v': 0.8605046201825178,
  'google w2v': 0.9643764122635148},
 'LogisticRegression with scaler': {'our w2v': 0.8634651466208176,
  'google w2v': 0.9726283137912249},
 'MultinomialNB': {'our w2v': 0.743971383152747,
  'google w2v': 0.9391554953079735},
 'KNeighborsClassifier': {'our w2v': 0.8303720163580858},
 'DecisionTreeClassifier': {'our w2v': 0.7537887600529158}}
```

KNeighborsClassifier

In [48]:
```
from sklearn.neighbors import KNeighborsClassifier

# KNC нужны значения, распределённые по нормальному распределению.
add_metrics_data("KNeighborsClassifier", "google w2v", classifier_pipel
```

	Категория	Точность
0	business	0.905660
1	entertainment	0.948905
2	politics	0.971223
3	sport	1.000000
4	tech	0.964539

Out[48]:
```
{'LogisticRegression': {'our w2v': 0.8605046201825178,
  'google w2v': 0.9643764122635148},
 'LogisticRegression with scaler': {'our w2v': 0.8634651466208176,
  'google w2v': 0.9726283137912249},
 'MultinomialNB': {'our w2v': 0.743971383152747,
  'google w2v': 0.9391554953079735},
 'KNeighborsClassifier': {'our w2v': 0.8303720163580858,
  'google w2v': 0.9580655031044948},
 'DecisionTreeClassifier': {'our w2v': 0.7537887600529158}}
```

DecisionTreeClassifier

In [49]:
```
from sklearn.tree import DecisionTreeClassifier

add_metrics_data("DecisionTreeClassifier", "google w2v", classifier_pip
```

	Категория	Точность
0	business	0.880503
1	entertainment	0.795620
2	politics	0.834532
3	sport	0.949686
4	tech	0.836879

Out[49]:
```
{'LogisticRegression': {'our w2v': 0.8605046201825178,
  'google w2v': 0.9643764122635148},
 'LogisticRegression with scaler': {'our w2v': 0.8634651466208176,
  'google w2v': 0.9726283137912249},
 'MultinomialNB': {'our w2v': 0.743971383152747,
  'google w2v': 0.9391554953079735},
 'KNeighborsClassifier': {'our w2v': 0.8303720163580858,
  'google w2v': 0.9580655031044948},
 'DecisionTreeClassifier': {'our w2v': 0.7537887600529158,
  'google w2v': 0.8594441847852641}}
```

TFIDF

Scaler не нужен (и его даже невозможно применить, ведь tfidf возвращает разреженную матрицу)

In [50]:
```
from sklearn.feature_extraction.text import TfidfVectorizer

tfidf = TfidfVectorizer(ngram_range=(1,3))
```

LogisticRegression

In [51]:
```
add_metrics_data("LogisticRegression", "tfidf", classifier_pipeline(tfi
```

	Категория	Точность
0	business	0.955975
1	entertainment	0.919708
2	politics	0.971223
3	sport	0.993711
4	tech	0.992908

Out[51]:
```
{'LogisticRegression': {'our w2v': 0.8605046201825178,
  'google w2v': 0.9643764122635148,
  'tfidf': 0.9667048773578898},
 'LogisticRegression with scaler': {'our w2v': 0.8634651466208176,
  'google w2v': 0.9726283137912249},
 'MultinomialNB': {'our w2v': 0.743971383152747,
  'google w2v': 0.9391554953079735},
 'KNeighborsClassifier': {'our w2v': 0.8303720163580858,
  'google w2v': 0.9580655031044948},
 'DecisionTreeClassifier': {'our w2v': 0.7537887600529158,
  'google w2v': 0.8594441847852641}}
```

MultinomialNaiveBayes

In [52]:
```
from sklearn.naive_bayes import MultinomialNB

# NB нужны неотрицательные значения.
add_metrics_data("MultinomialNB", "tfidf", classifier_pipeline(tfidf, M
```

	Категория	Точность
0	business	0.974843
1	entertainment	0.598540
2	politics	0.949640
3	sport	1.000000
4	tech	0.921986

Out[52]:
```
{'LogisticRegression': {'our w2v': 0.8605046201825178,
  'google w2v': 0.9643764122635148,
  'tfidf': 0.9667048773578898},
 'LogisticRegression with scaler': {'our w2v': 0.8634651466208176,
  'google w2v': 0.9726283137912249},
 'MultinomialNB': {'our w2v': 0.743971383152747,
  'google w2v': 0.9391554953079735,
  'tfidf': 0.8890018033307239},
 'KNeighborsClassifier': {'our w2v': 0.8303720163580858,
  'google w2v': 0.9580655031044948},
 'DecisionTreeClassifier': {'our w2v': 0.7537887600529158,
  'google w2v': 0.8594441847852641}}
```

KNeighborsClassifier

In [53]:
```python
from sklearn.neighbors import KNeighborsClassifier

# KNC нужны значения, распределённые по нормальному распределению.
add_metrics_data("KNeighborsClassifier", "tfidf", classifier_pipeline(t
```

	Категория	Точность
0	business	0.874214
1	entertainment	0.868613
2	politics	0.964029
3	sport	0.981132
4	tech	0.950355

Out[53]:
```
{'LogisticRegression': {'our w2v': 0.8605046201825178,
  'google w2v': 0.9643764122635148,
  'tfidf': 0.9667048773578898},
 'LogisticRegression with scaler': {'our w2v': 0.8634651466208176,
  'google w2v': 0.9726283137912249},
 'MultinomialNB': {'our w2v': 0.743971383152747,
  'google w2v': 0.9391554953079735,
  'tfidf': 0.8890018033307239},
 'KNeighborsClassifier': {'our w2v': 0.8303720163580858,
  'google w2v': 0.9580655031044948,
  'tfidf': 0.9276684875086623},
 'DecisionTreeClassifier': {'our w2v': 0.7537887600529158,
  'google w2v': 0.8594441847852641}}
```

DecisionTreeClassifier

In [54]:
```python
from sklearn.tree import DecisionTreeClassifier

add_metrics_data("DecisionTreeClassifier", "tfidf", classifier_pipeline
```

	Категория	Точность
0	business	0.748428
1	entertainment	0.656934
2	politics	0.719424
3	sport	0.880503
4	tech	0.709220

Out[54]:
```
{'LogisticRegression': {'our w2v': 0.8605046201825178,
  'google w2v': 0.9643764122635148,
  'tfidf': 0.9667048773578898},
 'LogisticRegression with scaler': {'our w2v': 0.8634651466208176,
  'google w2v': 0.9726283137912249},
 'MultinomialNB': {'our w2v': 0.743971383152747,
  'google w2v': 0.9391554953079735,
  'tfidf': 0.8890018033307239},
 'KNeighborsClassifier': {'our w2v': 0.8303720163580858,
  'google w2v': 0.9580655031044948,
  'tfidf': 0.9276684875086623},
 'DecisionTreeClassifier': {'our w2v': 0.7537887600529158,
  'google w2v': 0.8594441847852641,
  'tfidf': 0.7429018885534178}}
```

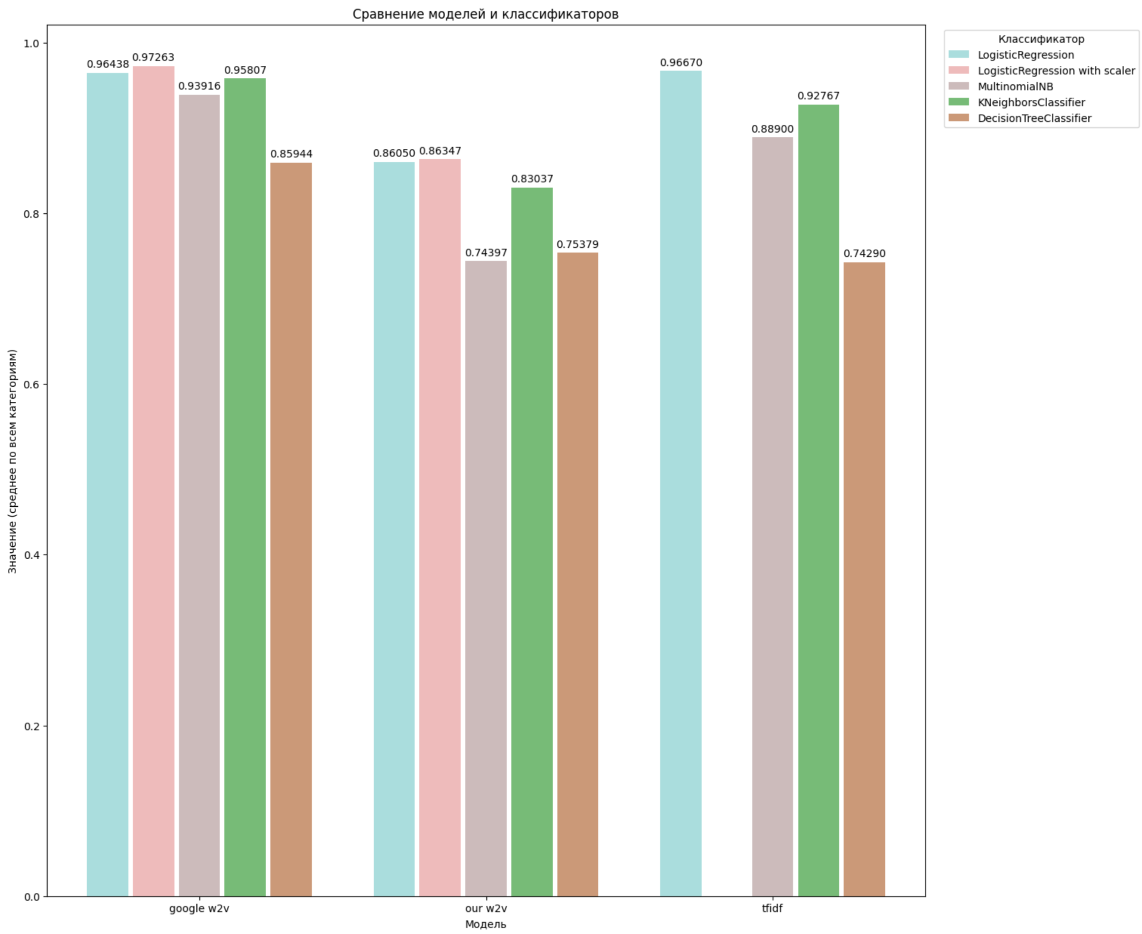
Сравнение результатов

```
In [62]: # Reloading editable packages.
%autoreload
from charts.main import grouped_bar_chart, get_metrics_grouped_bar_char

models_bar_chart = get_metrics_grouped_bar_chart(metrics_data)
models_bar_chart["plt"].title('Сравнение моделей и классификаторов')
models_bar_chart["plt"].xlabel('Модель')
models_bar_chart["plt"].ylabel('Значение (среднее по всем категориям)')

models_bar_chart["ax"].legend(title='Классификатор', bbox_to_anchor=(1.

models_bar_chart["plt"].show()
```



> У TFIDF нет "LogisticRegression with scaler", потому что scaler не нужен (и его невозможно применить для разреженной матрицы).

Вывод

Как видно по графику сравнения моделей и классификаторов, наиболее успешной оказалась связка word2vector, предобученная google, и логистической регрессии со scaler'ом.

В среднем, модель от google продемонстрировала самый высокий результат для любого классификатора. Единственное исключение - логистическая регрессия для tfidf. Это комбинация показала очень хороший результат, который лишь немного отстаёт от лучших значений google word2vector.

Так, видно, что в среднем tfidf достигает показателей лишь на 2-5 процентов хуже google2vector. Учитывая простоту алгоритма, это удивительный результат.

Обученная нами модель word2vec даёт в лучшем варианте 86%, что является вполне приемлемым. Однако в среднем у неё самые плохие результаты среди рассматриваемых моделей. Это довольно неожиданно, ведь обучение и тренировка на данных из одной предметной области должны были дать лучшее качество.

Пытаясь найти причины, первым приходит на ум недостаток данных: наш набор не очень спефичный и на несколько порядков меньше набора данных google. Но в то же время объём нашей выборки хоть и составляет несколько тысяч строк, в каждой из них есть ячейка с содержанием полноценной статьи, то есть количество токенов должно отвечать запросам word2vec.

Вероятно, первопричина кроется в качестве самого обучения: мы мало проводили работы с корпусом токенов. Сразу после разбиения текста с помощью `spacy` на токены, мы отправили модель обучаться. Вполне возможно, дополнительные оптимизации на уровне предложений и языка поспособствовали бы улучшению качества модели. Например, устранение стоп-слов или очень частых слов, несущих малый смысл в рамках нашей задачи (союзы, предлоги), из выборки. На это ещё больше намекает хороший результат TFIDF, ведь он эти оптимизации проводит "автоматически" ввиду особенностей формулы.

Таким образом, мы обучили три модели и сравнили показатели. Каждая из них даёт хорошие результаты в классификации, однако у всех моделей есть большой недостаток - неумение работать со словами, не присутствующими в выборке. Наименее заметно это в google word2vector, ведь в ней набор данных колоссальный, однако если стоит задача обрабатывать любые слова, лучше подойдёт принципиально модифицированная модель, например, fast2text.