Лабораторная работа 5

Предобработка и классификация текстовых данных.

Цель лабораторной работы: изучение методов предобработки и классификации текстовых данных.

Требования к отчету:

Отчет по лабораторной работе должен содержать:

титульный лист; описание задания; текст программы; экранные формы с примерами выполнения программы.

Задание:

Для произвольного предложения или текста решите следующие задачи:

Токенизация.

Частеречная разметка.

Лемматизация.

Выделение (распознавание) именованных сущностей.

Разбор предложения.

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 [52]:

```
!pip install natasha
```

```
Collecting natasha
  Downloading natasha-1.4.0-py3-none-any.whl (34.4 MB)
Collecting pymorphy2
  Downloading pymorphy2-0.9.1-py3-none-any.whl (55 kB)
Collecting slovnet>=0.3.0

WARNING: Ignoring invalid distribution -umpy (c:\programdata\anaconda3\lib\site-p
ackages)
WARNING: Ignoring invalid distribution -raitlets (c:\programdata\anaconda3\lib\si
te-packages)
WARNING: Ignoring invalid distribution - (c:\programdata\anaconda3\lib\site-packa
ges)
WARNING: Ignoring invalid distribution -umpy (c:\programdata\anaconda3\lib\site-p
ackages)
WARNING: Ignoring invalid distribution -raitlets (c:\programdata\anaconda3\lib\si
te-packages)
WARNING: Ignoring invalid distribution - (c:\programdata\anaconda3\lib\site-packa
ges)
WARNING: Ignoring invalid distribution -umpy (c:\programdata\anaconda3\lib\site-p
ackages)
WARNING: Ignoring invalid distribution -raitlets (c:\programdata\anaconda3\lib\si
te-packages)
WARNING: Ignoring invalid distribution - (c:\programdata\anaconda3\lib\site-packa
ges)
WARNING: Ignoring invalid distribution -umpy (c:\programdata\anaconda3\lib\site-p
ackages)
WARNING: Ignoring invalid distribution -raitlets (c:\programdata\anaconda3\lib\si
te-packages)
WARNING: Ignoring invalid distribution - (c:\programdata\anaconda3\lib\site-packa
ges)
WARNING: Ignoring invalid distribution -umpy (c:\programdata\anaconda3\lib\site-p
ackages)
WARNING: Ignoring invalid distribution -raitlets (c:\programdata\anaconda3\lib\si
te-packages)
WARNING: Ignoring invalid distribution - (c:\programdata\anaconda3\lib\site-packa
ges)
WARNING: Ignoring invalid distribution -umpy (c:\programdata\anaconda3\lib\site-p
ackages)
WARNING: Ignoring invalid distribution -raitlets (c:\programdata\anaconda3\lib\si
te-packages)
WARNING: Ignoring invalid distribution - (c:\programdata\anaconda3\lib\site-packa
ges)
WARNING: Ignoring invalid distribution -umpy (c:\programdata\anaconda3\lib\site-p
ackages)
WARNING: Ignoring invalid distribution -raitlets (c:\programdata\anaconda3\lib\si
te-packages)
WARNING: Ignoring invalid distribution - (c:\programdata\anaconda3\lib\site-packa
ges)
WARNING: Ignoring invalid distribution -umpy (c:\programdata\anaconda3\lib\site-p
ackages)
WARNING: Ignoring invalid distribution -raitlets (c:\programdata\anaconda3\lib\si
```

```
te-packages)
WARNING: Ignoring invalid distribution - (c:\programdata\anaconda3\lib\site-packa
ges)
WARNING: Ignoring invalid distribution -umpy (c:\programdata\anaconda3\lib\site-p
ackages)
WARNING: Ignoring invalid distribution -raitlets (c:\programdata\anaconda3\lib\si
te-packages)
WARNING: Ignoring invalid distribution - (c:\programdata\anaconda3\lib\site-packa
ges)
WARNING: Ignoring invalid distribution -umpy (c:\programdata\anaconda3\lib\site-p
ackages)
WARNING: Ignoring invalid distribution -raitlets (c:\programdata\anaconda3\lib\si
te-packages)
WARNING: Ignoring invalid distribution - (c:\programdata\anaconda3\lib\site-packa
ges)
WARNING: Ignoring invalid distribution -umpy (c:\programdata\anaconda3\lib\site-p
ackages)
WARNING: Ignoring invalid distribution -raitlets (c:\programdata\anaconda3\lib\si
te-packages)
WARNING: Ignoring invalid distribution - (c:\programdata\anaconda3\lib\site-packa
ges)
WARNING: Ignoring invalid distribution -umpy (c:\programdata\anaconda3\lib\site-p
ackages)
WARNING: Ignoring invalid distribution -raitlets (c:\programdata\anaconda3\lib\si
te-packages)
WARNING: Ignoring invalid distribution - (c:\programdata\anaconda3\lib\site-packa
ges)
WARNING: Ignoring invalid distribution -umpy (c:\programdata\anaconda3\lib\site-p
ackages)
WARNING: Ignoring invalid distribution -raitlets (c:\programdata\anaconda3\lib\si
te-packages)
WARNING: Ignoring invalid distribution - (c:\programdata\anaconda3\lib\site-packa
ges)
WARNING: Ignoring invalid distribution -umpy (c:\programdata\anaconda3\lib\site-p
ackages)
WARNING: Ignoring invalid distribution -raitlets (c:\programdata\anaconda3\lib\si
te-packages)
WARNING: Ignoring invalid distribution - (c:\programdata\anaconda3\lib\site-packa
ges)
WARNING: Ignoring invalid distribution -umpy (c:\programdata\anaconda3\lib\site-p
ackages)
WARNING: Ignoring invalid distribution -raitlets (c:\programdata\anaconda3\lib\si
te-packages)
WARNING: Ignoring invalid distribution - (c:\programdata\anaconda3\lib\site-packa
ges)


  Downloading slovnet-0.5.0-py3-none-any.whl (49 kB)
Collecting razdel>=0.5.0
  Downloading razdel-0.5.0-py3-none-any.whl (21 kB)
Collecting yargy>=0.14.0
  Downloading yargy-0.15.0-py3-none-any.whl (41 kB)
Collecting navec>=0.9.0
  Downloading navec-0.10.0-py3-none-any.whl (23 kB)
Collecting ipymarkup>=0.8.0
  Downloading ipymarkup-0.9.0-py3-none-any.whl (14 kB)
Requirement already satisfied: intervaltree>=3 in c:\programdata\anaconda3\lib\si
te-packages (from ipymarkup>=0.8.0->natasha) (3.1.0)
Requirement already satisfied: sortedcontainers<3.0,>=2.0 in c:\programdata\anaco
nda3\lib\site-packages (from intervaltree>=3->ipymarkup>=0.8.0->natasha) (2.4.0)
Requirement already satisfied: numpy in c:\users\asus\appdata\roaming\python\pyth
```

```
on39\site-packages (from navec>=0.9.0->natasha) (1.22.3)
Collecting pymorphy2-dicts-ru<3.0,>=2.4
  Downloading pymorphy2_dicts_ru-2.4.417127.4579844-py2.py3-none-any.whl (8.2 MB)
Collecting dawg-python>=0.7.1
  Downloading DAWG_Python-0.7.2-py2.py3-none-any.whl (11 kB)
Collecting docopt>=0.6
  Downloading docopt-0.6.2.tar.gz (25 kB)
Building wheels for collected packages: docopt
  Building wheel for docopt (setup.py): started
  Building wheel for docopt (setup.py): finished with status 'done'
  Created wheel for docopt: filename=docopt-0.6.2-py2.py3-none-any.whl size=13724
sha256=c2e2a10f3a935820cfcf485828805f3b4a953343d7838cb10a6f365936fd5e4b
  Stored in directory: c:\users\asus\appdata\local\pip\cache\wheels\70\4a\46\1309
fc853b8d395e60bafaf1b6df7845bdd82c95fd59dd8d2b
Successfully built docopt
Installing collected packages: pymorphy2-dicts-ru, docopt, dawg-python, razdel, p
ymorphy2, navec, yargy, slovnet, ipymarkup, natasha
Successfully installed dawg-python-0.7.2 docopt-0.6.2 ipymarkup-0.9.0 natasha-1.
4.0 navec-0.10.0 pymorphy2-0.9.1 pymorphy2-dicts-ru-2.4.417127.4579844 razdel-0.
5.0 slovnet-0.5.0 yargy-0.15.0
```

Text

In [53]:

```
text1 ='Моско́вский госуда́рственный техни́ческий университе́т
text2 ='杭州因风景秀丽，素有“人间天堂”的美誉。杭州得益于京杭大运河和通商口岸的便利，以及自身发达的
```

In [54]:

```
text1
```

Out[54]:

```
'Моско́вский госуда́рственный техни́ческий университе́т им. Н. Э. Ба́умана[a] (также известен как Ба́уманка, Ба́уманский, МГТУ, МВТУ) — российский национальный исследовательский университет, научный центр, особо ценный объект культурного наследия народов России'
```

In [55]:

```
from razdel import tokenize, sentenize
```

In [58]:

```
n_tok_text = list(tokenize(text1))
n_tok_text
```

Out[58]:

```
[Substring(0, 11, 'М о с к о́ в с к и й'),
 Substring(12, 28, 'г о с у д а́ р с т в е н н ы й'),
 Substring(29, 41, 'т е х н и́ ч е с к и й'),
 Substring(42, 54, 'у н и в е р с и т е́ т'),
 Substring(55, 57, 'и м'),
 Substring(57, 58, '.'),
 Substring(59, 60, 'Н'),
 Substring(60, 61, '.'),
 Substring(62, 63, 'Э'),
 Substring(63, 64, '.'),
 Substring(65, 73, 'Б а́ у м а н а'),
 Substring(73, 74, '['),
 Substring(74, 75, 'а'),
 Substring(75, 76, ']'),
 Substring(77, 78, '('),
 Substring(78, 83, 'т а к ж е'),
 Substring(84, 92, 'и з в е с т е н'),
 Substring(93, 96, 'к а к'),
 Substring(97, 106, 'Б а́ у м а н к а'),
 Substring(106, 107, ','),
 Substring(108, 119, 'Б а́ у м а н с к и й'),
 Substring(119, 120, ','),
 Substring(121, 125, 'М Г Т У'),
 Substring(125, 126, ','),
 Substring(127, 131, 'М В Т У'),
 Substring(131, 132, ')'),
 Substring(133, 134, '—'),
 Substring(135, 145, 'р о с с и й с к и й'),
 Substring(146, 158, 'н а ц и о н а л ь н ы й'),
 Substring(159, 176, 'и с с л е д о в а т е л ь с к и й'),
 Substring(177, 188, 'у н и в е р с и т е т'),
 Substring(188, 189, ','),
 Substring(190, 197, 'н а у ч н ы й'),
 Substring(198, 203, 'ц е н т р'),
 Substring(203, 204, ','),
 Substring(205, 210, 'о с о б о'),
 Substring(211, 217, 'ц е н н ы й'),
 Substring(218, 224, 'о б ъ е к т'),
 Substring(225, 236, 'к у л ь т у р н о г о'),
 Substring(237, 245, 'н а с л е д и я'),
 Substring(246, 253, 'н а р о д о в'),
 Substring(254, 260, 'Р о с с и и')]
```

In [60]:

```
n_sen_text = list(sentenize(text1))
n_sen_text
```

Out[60]:

```
[Substring(0,
           260,
           'Моско́вский госуда́рственный техни́ческий
университе́т им. Н. Э. Ба́умана[a] (также известен к
ак Ба́уманка, Ба́уманский, МГТУ, МВТУ) — российски
й национальный исследовательский университе
т, научный центр, особо ценный объект культурн
ого наследия народов России')]
```

In [61]:

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

Out[61]:

```
(['Моско́вский госуда́рственный техни́ческий унив
ерсите́т им. Н. Э. Ба́умана[a] (также известен как Б
а́уманка, Ба́уманский, МГТУ, МВТУ) — российский на
циональный исследовательский университет, на
учный центр, особо ценный объект культурного
наследия народов России'],
 1)
```

In [62]:

```
# Этот вариант токенизации нужен для последующей обраб
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 [64]:

```
n_sen_chunk = n_sentenize(text1)
n_sen_chunk
```

Out[64]:

```
[['М о с к о́ в с к и й',
  'г о с у д а́ р с т в е н н ы й',
  'т е х н и́ ч е с к и й',
  'у н и в е р с и т е́ т',
  'и м',
  ',',
  '.',
  'Н',
  ',',
  '.',
  'Э',
  ',',
  '.',
  'Б а́ у м а н а',
  '[',
  'а',
  ']',
  '(',
  'т а к ж е',
  'и з в е с т е н',
  'к а к',
  'Б а́ у м а н к а',
  ',',
  ',',
  'Б а́ у м а н с к и й',
  ',',
  ',',
  'М Г Т У',
  ',',
  ',',
  'М В Т У',
  ')',
  '—',
  'р о с с и й с к и й',
  'н а ц и о н а л ь н ы й',
  'и с с л е д о в а т е л ь с к и й',
  'у н и в е р с и т е т',
  ',',
  ',',
  'н а у ч н ы й',
  'ц е н т р',
  ',',
  ',',
  'о с о б о',
  'ц е н н ы й',
  'о б ъ е к т',
  'к у л ь т у р н о г о',
  'н а с л е д и я',
  'н а р о д о в',
  'Р о с с и и']]
```

In [65]:

```
n_sen_chunk_2 = n_sentenize(text2)
n_sen_chunk_2
```

Out[65]:

```
[['杭州因风景秀丽，素有',
  '“',
  '人间天堂',
  '”',
  '的美誉。杭州得益于京杭大运河和通商口岸的便利，以及自身发达的丝绸和粮食产业，历史上曾是重要的商业集散中心。新世纪以来，随着阿里巴巴等高科技企业的带动，互联网经济成为杭州新的经济增长点。']]
```

Частеречная разметка

In [66]:

```
from navec import Navec
from slovnet import Morph
```

In [ ]:

```

```

In [70]:

```
# Файл необходимо скачать по ссылке https://github.com/natasha/navec#do
navec = Navec.load(r'C:\Users\asus\Desktop\iu5\MMO\lab5\navec_news_v1_1B_250K_300d_100q.tar')
```

In [71]:

```
# Файл необходимо скачать по ссылке https://github.com/natasha/slovnet#
n_morph = Morph.load(r'C:\Users\asus\Desktop\iu5\MMO\lab5\slovnet_morph_news_v1.tar', batch_size=4)
```

In [72]:

```
morph_res = n_morph.navec(navec)
```

In [73]:

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

In [74]:

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

```
Моско́вский - PROPN|Animacy=Anim|Case=Nom|Gender=Masc|Number=Sing
государ́ственный - X|Foreign=Yes
технический - ADJ|Case=Gen|Degree=Pos|Gender=Masc|Number=Sing
университе́т - NOUN|Animacy=Inan|Case=Gen|Gender=Masc|Number=Sing
им - PRON|Case=Ins|Gender=Masc|Number=Sing|Person=3
. - PUNCT
Н - PROPN|Animacy=Anim|Case=Gen|Gender=Masc|Number=Sing
. - PUNCT
Э - PROPN|Animacy=Anim|Case=Gen|Gender=Masc|Number=Sing
. - PUNCT
Ба́умана - PROPN|Animacy=Anim|Case=Gen|Gender=Masc|Number=Sing
[ - PUNCT
a - X|Foreign=Yes
] - PUNCT
( - PUNCT
также - ADV|Degree=Pos
известен - ADJ|Degree=Pos|Gender=Masc|Number=Sing|Variant=Short
как - SCONJ
Ба́уманка - PROPN|Animacy=Anim|Case=Nom|Gender=Masc|Number=Sing
, - PUNCT
Ба́уманский - PROPN|Animacy=Inan|Case=Nom|Gender=Masc|Number=Sing
, - PUNCT
МГТУ - PROPN|Animacy=Inan|Case=Gen|Gender=Neut|Number=Sing
, - PUNCT
МВТУ - PROPN|Animacy=Inan|Case=Gen|Gender=Masc|Number=Sing
) - PUNCT
— - PUNCT
российский - ADJ|Case=Nom|Degree=Pos|Gender=Masc|Number=Sing
национальный - ADJ|Case=Nom|Degree=Pos|Gender=Masc|Number=Sing
исследовательский - ADJ|Case=Nom|Degree=Pos|Gender=Masc|Number=Sing
университет - NOUN|Animacy=Inan|Case=Nom|Gender=Masc|Number=Sing
, - PUNCT
научный - ADJ|Case=Nom|Degree=Pos|Gender=Masc|Number=Sing
центр - NOUN|Animacy=Inan|Case=Nom|Gender=Masc|Number=Sing
, - PUNCT
особо - ADV|Degree=Pos
ценный - ADJ|Case=Nom|Degree=Pos|Gender=Masc|Number=Sing
объект - NOUN|Animacy=Inan|Case=Nom|Gender=Masc|Number=Sing
культурного - ADJ|Case=Gen|Degree=Pos|Gender=Neut|Number=Sing
наследия - NOUN|Animacy=Inan|Case=Gen|Gender=Neut|Number=Sing
народов - NOUN|Animacy=Inan|Case=Gen|Gender=Masc|Number=Plur
России - PROPN|Animacy=Inan|Case=Gen|Gender=Fem|Number=Sing
```

Out[74]:

```
[None]
```

In [75]:

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

```
杭州因风景秀丽，素有 - PUNCT
“ - PUNCT
人间天堂 - PUNCT
” - PUNCT
的美誉。杭州得益于京杭大运河和通商口岸的便利，以及自身发达的丝绸和粮食产业，历史上曾
是重要的商业集散中心。新世纪以来，随着阿里巴巴等高科技企业的带动，互联网经济成为杭州
新的经济增长点。 - PUNCT
```

Out[75]:

```
[None]
```

In [76]:

```
from natasha import Doc, Segmenter, NewsEmbedding, NewsMorphTagger, MorphVocab
```

In [83]:

```
#词性还原
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 [87]:

```
n_doc = n_lemmatize(text1)
{_.text: _.lemma for _ in n_doc.tokens}
```

Out[87]:

```
{'Москóвский': 'москóвский',
 'государственный': 'государственный',
 'технический': 'технический',
 'университéт': 'университéт',
 'им': 'он',
 '.': '.',
 'Н': 'н',
 'Э': 'э',
 'Баумана': 'бáуман',
 '[': '[',
 'а': 'а',
 ']': ']',
 '(': '(',
 'также': 'также',
 'известен': 'известный',
 'как': 'как',
 'Бáуманка': 'бáуманка',
 ',': ',',
 'Бáуманский': 'бáуманский',
 'МГТУ': 'мгту',
 'МВТУ': 'мвт',
 ')': ')',
 '—': '—',
 'российский': 'российский',
 'национальный': 'национальный',
 'исследовательский': 'исследовательский',
 'университет': 'университет',
 'научный': 'научный',
 'центр': 'центр',
 'особо': 'особо',
 'ценный': 'ценный',
 'объект': 'объект',
 'культурного': 'культурный',
 'наследия': 'наследие',
 'народов': 'народ',
 'России': 'россия'}
```

In [88]:

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

Out[88]:

{'杭州因风景秀丽，素有': '杭州因风景秀丽，素有',
 '“': '“',
 '人间天堂': '人间天堂',
 '”': '”',
 '的美誉。杭州得益于京杭大运河和通商口岸的便利，以及自身发达的丝绸和粮食产业，历史上曾是重要的商业集散中心。新世纪以来，随着阿里巴巴等高科技企业的带动，互联网经济成为杭州新的经济增长点。': '的美誉。杭州得益于京杭大运河和通商口岸的便利，以及自身发达的丝绸和粮食产业，历史上曾是重要的商业集散中心。新世纪以来，随着阿里巴巴等高科技企业的带动，互联网经济成为杭州新的经济增长点。'}

Выделение (распознавание) именованных сущностей

In [90]:

```
#name entity tagging (命名实体标记)
from slovnet import NER
from ipymarkup import show_span_ascii_markup as show_markup
```

In [92]:

```
ner = NER.load(r'C:\Users\asus\Desktop\iu5\MMO\lab5\slovnet_ner_news_v1.tar')
```

In [93]:

```
ner_res = ner.navec(navec)
```

In [94]:

```
markup_ner = ner(text2)
markup_ner
```

Out[94]:

```
SpanMarkup(
    text='杭州因风景秀丽，素有“人间天堂”的美誉。杭州得益于京杭大运河和通商口岸的便利，以及自身发达的丝绸和粮食产业，历史上曾是重要的商业集散中心。新世纪以来，随着阿里巴巴等高科技企业的带动，互联网经济成为杭州新的经济增长点。',
    spans=[]
)
```

In [95]:

```
show_markup(markup_ner.text, markup_ner.spans)
```

杭州因风景秀丽，素有“人间天堂”的美誉。杭州得益于京杭大运河和通商口岸的便利，以及自身发达的丝绸和粮食产业，历史上曾是重要的商业集散中心。新
世纪以来，随着阿里巴巴等高科技企业的带动，互联网经济成为杭州新的经济增长点。

Разбор предложения（语法解析）

In [99]:

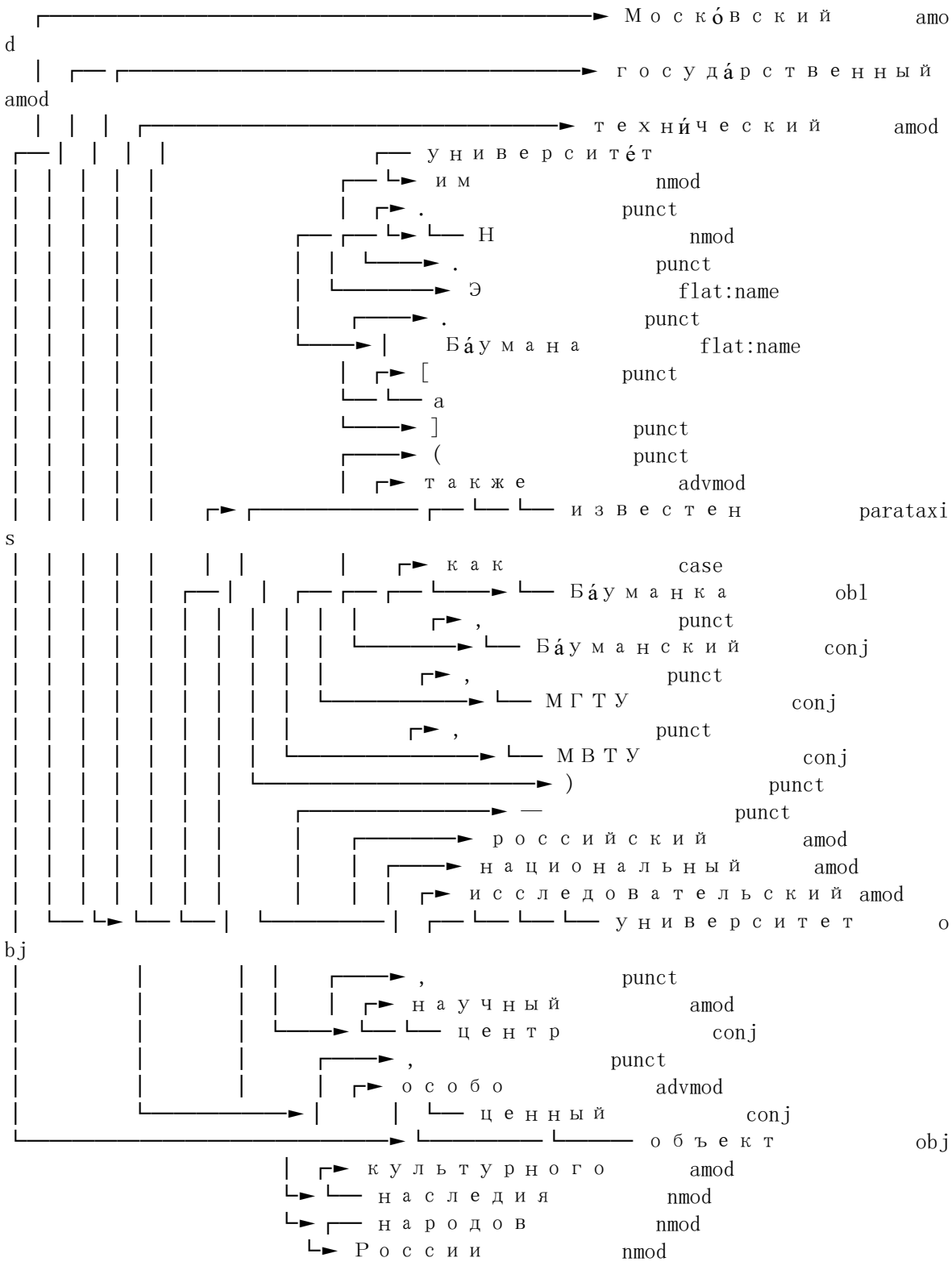
```
from natasha import NewsSyntaxParser
```

In [100]:

```
emb = NewsEmbedding()
syntax_parser = NewsSyntaxParser(emb)
```

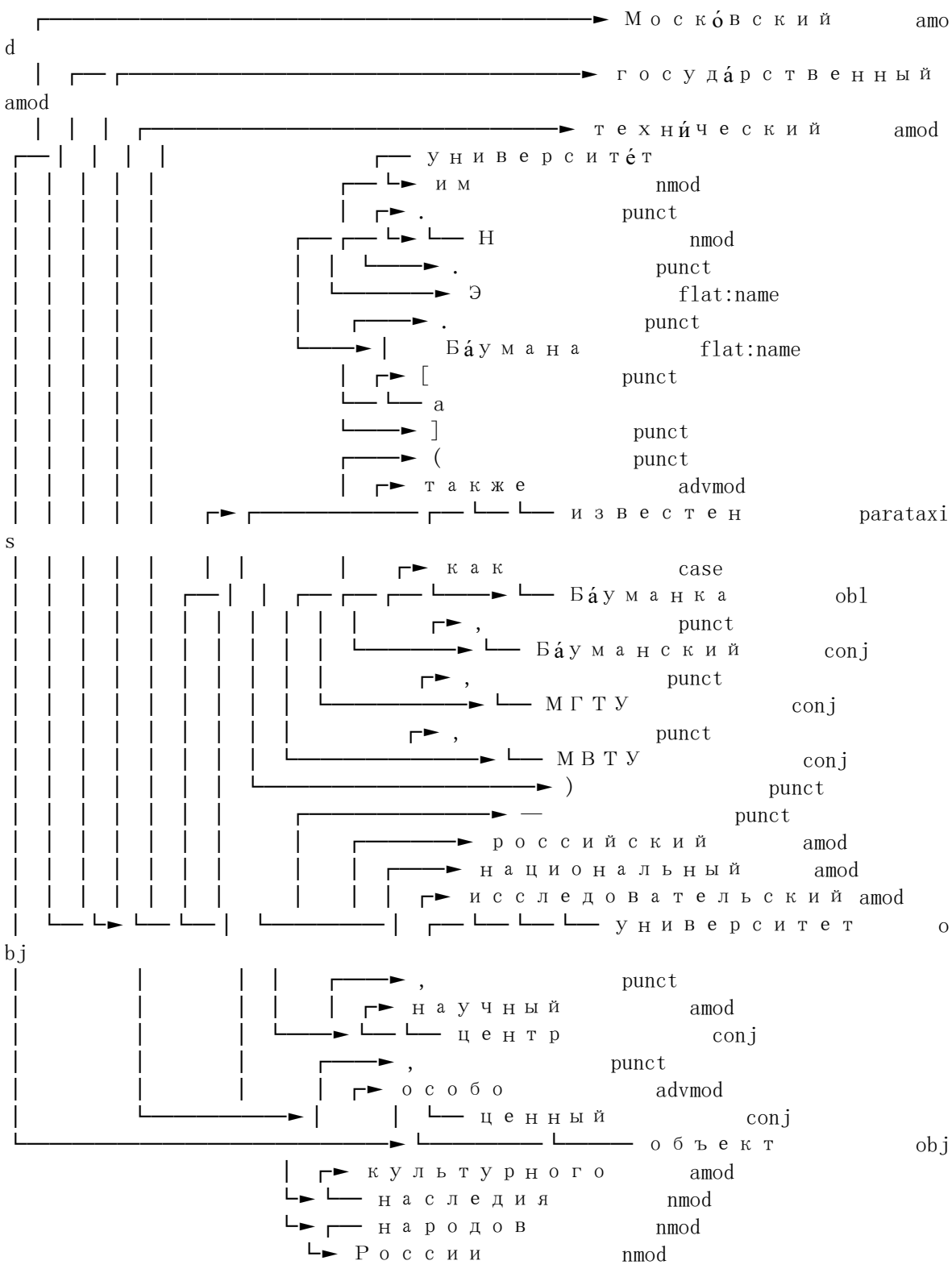
In [101]:

```
n_doc.parse_syntax(syntax_parser)
n_doc.sents[0].syntax.print()
```



In [103]:

```
n_doc.parse_syntax(syntax_parser)
n_doc.sents[0].syntax.print()
```



In [104]:

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

```
        杭州因风景秀丽，素有
     ┌► “
punct
 ┌─ ┌─ └─ 人间天堂
 │  └──► ”
punct
 └─────► 的美誉。杭州得益于京杭大运河和通商口岸的便利，以及自身发达的丝绸和粮食
产业，历史上曾是重要的商业集散中心。新世纪以来，随着阿里巴巴等高科技企业的带动，互联
网经济成为杭州新的经济增长点。 punct
```

Векторизация текста на основе модели "мешка слов" (bag of words)

In [105]:

```
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_report
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_error, median
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
import urllib.request

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

In [106]:

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

In [107]:

```
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 [111]:

```
vocabVect = CountVectorizer()
vocabVect.fit(data)
corpusVocab = vocabVect.vocabulary_
print('Количество сформированных признаков - {}'.format(len(corpusV
```

```
Количество сформированных признаков - 33448
```

In [112]:

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

```
nrmendel=22213
unix=31462
amherst=5287
edu=12444
nathaniel=21624
mendell=20477
subject=29220
re=25369
bike=6898
```

Использование класса CountVectorizer(使用 CountVectorizer 类)

In [113]:

```
test_features = vocabVect.transform(data)
test_features
```

Out[113]:

```
<2380x33448 sparse matrix of type '<class 'numpy.int64'>'
        with 335176 stored elements in Compressed Sparse Row format>
```

In [114]:

```
test_features.todense()
```

Out[114]:

```
matrix([[0, 0, 0, ..., 0, 0, 0],
        [0, 0, 0, ..., 0, 0, 0],
        [0, 0, 0, ..., 0, 0, 0],
        ...,
        [2, 0, 0, ..., 0, 0, 0],
        [0, 0, 0, ..., 0, 0, 0],
        [0, 0, 0, ..., 0, 0, 0]], dtype=int64)
```

In [115]:

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

Out[115]:

```
33448
```

In [116]:

```
print([i for i in test_features.todense()[0].getA1() if i>0])
```

```
[1, 1, 1, 1, 1, 2, 1, 1, 1, 2, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2, 1, 1, 2, 1, 1, 1,
1, 1, 1, 1, 1, 2, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
3, 2]
```

In [117]:

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

Out[117]:

```
['00',
 '000',
 '0000',
 '0000000004',
 '0000000005',
 '0000000667',
 '0000001200',
 '0001',
 '00014',
 '0002']
```

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

In [118]:

```
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'], scoring='ac
            print('Векторизация - {}'.format(v))
            print('Модель для классификации - {}'.format(c))
            print('Accuracy = {}'.format(score))
            print('============================')
```

In [119]:

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

```
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\linear_model\_logistic.py:763: Co
nvergenceWarning: 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 (https://scikit-lear
n.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 (h
ttps://scikit-learn.org/stable/modules/linear_model.html#logistic-regression)
  n_iter_i = _check_optimize_result(
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\linear_model\_logistic.py:763: Co
nvergenceWarning: 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 (https://scikit-lear
n.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 (h
ttps://scikit-learn.org/stable/modules/linear_model.html#logistic-regression)
  n_iter_i = _check_optimize_result(
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\linear_model\_logistic.py:763: Co
nvergenceWarning: 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 (https://scikit-lear
n.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 (h
ttps://scikit-learn.org/stable/modules/linear_model.html#logistic-regression)
  n_iter_i = _check_optimize_result(


Векторизация - CountVectorizer(vocabulary={'00': 0, '000': 1, '0000': 2,
'0000000004': 3,
                            '0000000005': 4, '0000000667': 5, '0000001200': 6,
                            '0001': 7, '00014': 8, '0002': 9, '0003': 10,
                            '0005111312': 11, '0005111312na1em': 12,
                            '00072': 13, '000851': 14, '000rpm': 15,
                            '000th': 16, '001': 17, '0010': 18, '001004': 19,
                            '0011': 20, '001211': 21, '0013': 22, '001642': 23,
                            '001813': 24, '002': 25, '002222': 26,
                            '002251w': 27, '0023': 28, '002937': 29, ...})
Модель для классификации - LogisticRegression(C=3.0)
Accuracy = 0.9382336841146768
===========================
Векторизация - CountVectorizer(vocabulary={'00': 0, '000': 1, '0000': 2,
'0000000004': 3,
                            '0000000005': 4, '0000000667': 5, '0000001200': 6,
                            '0001': 7, '00014': 8, '0002': 9, '0003': 10,
                            '0005111312': 11, '0005111312na1em': 12,
                            '00072': 13, '000851': 14, '000rpm': 15,
                            '000th': 16, '001': 17, '0010': 18, '001004': 19,
```

```
                                'OOll': 20, 'OOl2ll': 21, 'OOl3': 22, 'OOl642': 23,
                                'OOl8l3': 24, 'OO2': 25, 'O02222': 26,
                                'O02251w': 27, 'O023': 28, 'O02937': 29, ...})
Модель для классификации - LinearSVC()
Accuracy = 0.9453742497059174
============================
Векторизация - CountVectorizer(vocabulary={'00': 0, '000': 1, '0000': 2,
'0000000004': 3,
                                '0000000005': 4, '0000000667': 5, '0000001200': 6,
                                '0001': 7, '00014': 8, '0002': 9, '0003': 10,
                                '0005111312': 11, '0005111312na1em': 12,
                                '00072': 13, '000851': 14, '000rpm': 15,
                                '000th': 16, '001': 17, '0010': 18, '001004': 19,
                                '0011': 20, '001211': 21, '0013': 22, '001642': 23,
                                '001813': 24, '002': 25, '002222': 26,
                                '002251w': 27, '0023': 28, '002937': 29, ...})
Модель для классификации - KNeighborsClassifier()
Accuracy = 0.6655358653541747
============================
```

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

In [120]:

```
X_train, X_test, y_train, y_test = train_test_split(newsgroups['data'], newsgroups['target'], test_s
```

In [121]:

```
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 [122]:

```
sentiment(CountVectorizer(), LinearSVC())
```

```
Метка      Accuracy
0       0.9290322580645162
1       0.9675090252707581
2       0.9026845637583892
3       0.9245901639344263
```

Работа с векторными представлениями слов с использованием word2vec

In [159]:

```
!pip install --user gensim
import gensim
from gensim.models import word2vec
```

```
Collecting gensim
  Using cached gensim-4.2.0-cp39-cp39-win_amd64.whl (23.9 MB)
Requirement already satisfied: smart-open>=1.8.1 in c:\programdata\anaconda3\lib\sit
e-packages (from gensim) (5.2.1)
Requirement already satisfied: numpy>=1.17.0 in c:\users\asus\appdata\roaming\python
\python39\site-packages (from gensim) (1.22.3)
Collecting Cython==0.29.28
  Using cached Cython-0.29.28-py2.py3-none-any.whl (983 kB)
Requirement already satisfied: scipy>=0.18.1 in c:\programdata\anaconda3\lib\site-pa
ckages (from gensim) (1.7.1)
Installing collected packages: Cython, gensim
Successfully installed Cython-0.29.28 gensim-4.2.0

WARNING: Ignoring invalid distribution -umpy (c:\programdata\anaconda3\lib\site-pack
ages)
WARNING: Ignoring invalid distribution -raitlets (c:\programdata\anaconda3\lib\site-
packages)
WARNING: Ignoring invalid distribution - (c:\programdata\anaconda3\lib\site-package
s)
WARNING: Ignoring invalid distribution -umpy (c:\programdata\anaconda3\lib\site-pack
ages)
WARNING: Ignoring invalid distribution -raitlets (c:\programdata\anaconda3\lib\site-
packages)
WARNING: Ignoring invalid distribution - (c:\programdata\anaconda3\lib\site-package
s)
WARNING: Ignoring invalid distribution -umpy (c:\programdata\anaconda3\lib\site-pack
ages)
WARNING: Ignoring invalid distribution -raitlets (c:\programdata\anaconda3\lib\site-
packages)
WARNING: Ignoring invalid distribution - (c:\programdata\anaconda3\lib\site-package
s)
  WARNING: The scripts cygdb.exe, cython.exe and cythonize.exe are installed in
'C:\Users\asus\AppData\Roaming\Python\Python39\Scripts' which is not on PATH.
  Consider adding this directory to PATH or, if you prefer to suppress this warning,
use --no-warn-script-location.
WARNING: Ignoring invalid distribution -umpy (c:\programdata\anaconda3\lib\site-pack
ages)
WARNING: Ignoring invalid distribution -raitlets (c:\programdata\anaconda3\lib\site-
packages)
WARNING: Ignoring invalid distribution - (c:\programdata\anaconda3\lib\site-package
s)
WARNING: Ignoring invalid distribution -umpy (c:\programdata\anaconda3\lib\site-pack
ages)
WARNING: Ignoring invalid distribution -raitlets (c:\programdata\anaconda3\lib\site-
packages)
WARNING: Ignoring invalid distribution - (c:\programdata\anaconda3\lib\site-package
s)
WARNING: Ignoring invalid distribution -umpy (c:\programdata\anaconda3\lib\site-pack
ages)
WARNING: Ignoring invalid distribution -raitlets (c:\programdata\anaconda3\lib\site-
packages)
WARNING: Ignoring invalid distribution - (c:\programdata\anaconda3\lib\site-package
s)
```

In [160]:

```
urllib.request.urlretrieve("http://rusvectores.org/static/models/rusvectores2/ruscorpora_mystem_cbow
```

Out[160]:

```
('ruscorpora_mystem_cbow_300_2_2015.bin.gz',
 <http.client.HTTPMessage at 0x254e2cceaf0>)
```

In [161]:

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

In [162]:

```
model = gensim.models.KeyedVectors.load_word2vec_format(model_path, binary=True)
```

In [163]:

```
words = ['холод_S', 'мороз_S', 'береза_S', 'сосна_S']
```

In [164]:

```
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 - холод_S
5 ближайших соседей слова:
стужа_S => 0.7676383852958679
сырость_S => 0.6338975429534912
жара_S => 0.6089427471160889
мороз_S => 0.5890367031097412
озноб_S => 0.5776054859161377

СЛОВО - мороз_S
5 ближайших соседей слова:
стужа_S => 0.6425479650497437
морозец_S => 0.5947279930114746
холод_S => 0.5890367031097412
жара_S => 0.5522176623344421
снегопад_S => 0.5083199143409729

СЛОВО - береза_S
5 ближайших соседей слова:
сосна_S => 0.7943247556686401
тополь_S => 0.7562226057052612
дуб_S => 0.7440178394317627
дерево_S => 0.7373415231704712
клен_S => 0.7105200290679932

СЛОВО - сосна_S
5 ближайших соседей слова:
береза_S => 0.7943247556686401
дерево_S => 0.7581434845924377
лиственница_S => 0.747814953327179
дуб_S => 0.7412480711936951
ель_S => 0.7363824248313904
```

Находим близость между словами и строим аналогии

In [165]:

```
print(model.most_similar(positive=['холод_S', 'стужа_S'], negative=['мороз_S']))
```

```
[('сырость_S', 0.5040211081504822), ('стылость_S', 0.4633612930774688
7), ('голод_S', 0.4604816436767578), ('зной_S', 0.45904627442359924), ('с
кука_S', 0.4489358067512512), ('жара_S', 0.44645121693611145), ('усталл
ость_S', 0.4218570291996002), ('озноб_S', 0.41469818353652954), ('духо
та_S', 0.4099087715148926), ('неуют_S', 0.40298789739608765)]
```

Обучим word2vec на наборе данных "fetch_20newsgroups"

In [166]:

```
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
[nltk_data]     C:\Users\asus\AppData\Roaming\nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
```

Out[166]:

```
True
```

In [167]:

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

In [168]:

```
# 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 [169]:

```
corpus[:5]
```

Out[169]:

```
[['nrmendel',
  'unix',
  'amherst',
  'edu',
  'nathaniel',
  'mendell',
  'subject',
  'bike',
  'advice',
  'organization',
  'amherst',
  'college',
  'x',
  'newsreader',
  'tin',
  'version',
  'pl',
  'lines',
```

In [170]:

```
%time model_imdb = word2vec.Word2Vec(corpus, workers=4, min_count=10, window=10, sample=1e-3)
```

```
Wall time: 1.99 s
```

In [171]:

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

```
[('want', 0.98796147108078), ('work', 0.9793075919151306), ('etc', 0.978960871696472
2), ('used', 0.9754022359848022), ('using', 0.9715811014175415)]
```

In [172]:

```
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 [173]:

```
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 [174]:

```
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 [175]:

```
# 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 [176]:

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

```
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\linear_model\_logistic.py:763: Co
nvergenceWarning: 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 (https://scikit-lear
n.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 (h
ttps://scikit-learn.org/stable/modules/linear_model.html#logistic-regression)
  n_iter_i = _check_optimize_result(
```

Метка	Accuracy
0	0.8640350877192983
1	0.9320388349514563
2	0.8256880733944955
3	0.7587719298245614

In [ ]:

In [ ]:

In [ ]:

In [ ]:

Type *Markdown* and LaTeX: α^2

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

```

```

In [ ]:

```

```