Министерство науки и высшего образования Российской Федерации Федеральное государственное бюджетное образовательное учреждение высшего образования «Московский государственный технический университет имени Н.Э. Баумана (национальный исследовательский университет)» (МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ Информатика и системы управления и искусственный интеллект

КАФЕДРА Системы обработки информации и управления

Лабораторная работа №8

По курсу

«Методы машинного обучения»

«Предобработка текста»

Студент _____ **Костарев А. П.**
подпись, дата *фамилия, и .о.*

Преподаватель _____ **Гапанюк Ю. Е.**
подпись, дата *фамилия, и .о*

2024 г.

Цель работы: изучение методов предобработки текстов.

Задание

Для произвольного предложения или текста решите следующие задачи:

- Токенизация.
- Частеречная разметка.
- Лемматизация.
- Выделение (распознавание) именованных сущностей.
- Разбор предложения.

Выполнение задания

Токенизация

```
text1 = 'Адаптивная вёрстка меняет дизайн страницы в зависимости от поведения пользователя, платформы, размера экрана и ориентации девайса и является неотъемлемой частью современной веб-разработки.'
text2 = 'Покупатели могут получать заказы в пунктах выдачи заказов, в почтоматах (преимуществом этого варианта является экономия на оплате труда) или заказы могут доставляться им непосредственно до дома (или места работы) курьерами.'
text3 = 'Содержит ли это излучение информацию об их внутренней структуре, как это предполагает дуальность тяготение-калибровочная инвариантность, или нет, как следует из оригинального расчёта Хокинга?'
```

```
import nltk
from nltk.tokenize import punkt
nltk.download('punkt')
```

```
from nltk import tokenize
dir(tokenize)[:18]
```

```
['BlanklineTokenizer',
 'LegalitySyllableTokenizer',
 'LineTokenizer',
 'MWETokenizer',
 'NLTKWordTokenizer',
 'PunktSentenceTokenizer',
 'RegexpTokenizer',
 'ReppTokenizer',
 'SExprTokenizer',
 'SpaceTokenizer',
 'StanfordSegmenter',
 'SyllableTokenizer',
 'TabTokenizer',
 'TextTilingTokenizer',
 'ToktokTokenizer',
 'TreebankWordDetokenizer',
 'TreebankWordTokenizer',
 'TweetTokenizer']
```

```
nltk_tk_1 = nltk.WordPunctTokenizer()
nltk_tk_1.tokenize(text1)
```

```
['Адаптивная',
 'вёрстка',
 'меняет',
 'дизайн',
 'страницы',
 'в',
 'зависимости',
 'от',
 'поведения',
 'пользователя',
 ',',
 'платформы',
 ',',
 'размера',
 'экрана',
 'и',
 'ориентации',
 'девайса',
 'и',
 'является',
 'неотъемлемой',
 'частью',
 'современной',
 'веб',
 '-',
 'разработки',
 '.']
```

```
] # Токенизация по предложениям
  nltk_tk_sents = nltk.tokenize.sent_tokenize(text1)
  print(len(nltk_tk_sents))
  nltk_tk_sents
```

```
1
['Адаптивная вёрстка меняет дизайн страницы в зависимости от поведения пользователя, платформы, размера экрана и ориентации девайса и является неотъемлемой частью современной веб-разработки.']
```

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

```
n_tok_text1 = list(tokenize(text1))
n_tok_text1
```

```
[Substring(0, 10, 'Адаптивная'),
 Substring(11, 18, 'вёрстка'),
 Substring(19, 25, 'меняет'),
 Substring(26, 32, 'дизайн'),
 Substring(33, 41, 'страницы'),
 Substring(42, 43, 'в'),
 Substring(44, 55, 'зависимости'),
 Substring(56, 58, 'от'),
 Substring(59, 68, 'поведения'),
 Substring(69, 81, 'пользователя'),
 Substring(81, 82, ','),
 Substring(83, 92, 'платформы'),
 Substring(92, 93, ','),
 Substring(94, 101, 'размера'),
 Substring(102, 108, 'экрана'),
 Substring(109, 110, 'и'),
 Substring(111, 121, 'ориентации'),
 Substring(122, 129, 'девайса'),
 Substring(130, 131, 'и'),
 Substring(132, 140, 'является'),
 Substring(141, 153, 'неотъемлемой'),
 Substring(154, 160, 'частью'),
 Substring(161, 172, 'современной'),
 Substring(173, 187, 'веб-разработки'),
 Substring(187, 188, '.')]
```

```
[_.text for _ in n_tok_text1]
```

```
['Адаптивная',
 'вёрстка',
 'меняет',
 'дизайн',
 'страницы',
 'в',
 'зависимости',
 'от',
 'поведения',
 'пользователя',
 ',',
 'платформы',
 ',',
 'размера',
 'экрана',
 'и',
 'ориентации',
 'девайса',
 'и',
 'является',
 'неотъемлемой',
 'частью',
 'современной',
 'веб-разработки',
 '.']
```

```
n_sen_text1 = list(sentenize(text1))
n_sen_text1
```

[Substring(0,

188,

'Адаптивная вёрстка меняет дизайн страницы в зависимости от поведения пользователя, платформы, размера экрана и ориентации девайса и является неотъемлемой частью современной веб-разработки.')]

```
[_.text for _ in n_sen_text1], len([_.text for _ in n_sen_text1])
```

(['Адаптивная вёрстка меняет дизайн страницы в зависимости от поведения пользователя, платформы, размера экрана и ориентации девайса и является неотъемлемой частью современной веб-разработки.'],

1)

```
# 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
```

```
n_sen_chunk_1 = n_sentenize(text1)
n_sen_chunk_1
```

```
[['Адаптивная',
  'вёрстка',
  'меняет',
  'дизайн',
  'страницы',
  'в',
  'зависимости',
  'от',
  'поведения',
  'пользователя',
  ',',
  'платформы',
  ',',
  'размера',
  'экрана',
  'и',
  'ориентации',
  'девайса',
  'и',
  'является',
  'неотъемлемой',
  'частью',
  'современной',
  'веб-разработки',
  '.']]
```

```
[31] n_sen_chunk_2 = n_sentenize(text2)
     n_sen_chunk_2
```

```
[['Покупатели',
  'могут',
  'получать',
  'заказы',
  'в',
  'пунктах',
  'выдачи',
  'заказов',
  ',',
  'в',
  'почтоматах',
  '(',
  'преимуществом',
  'этого',
  'варианта',
  'является',
  'экономия',
  'на',
  'оплате',
  'труда',
  ')',
  'или',
  'заказы',
  'могут',
  'доставляться',
  'им',
  'непосредственно',
  'до',
  'дома',
  '(',
  'или',
  'места',
  'работы',
  ')',
  'курьерами',
  '.']]
```

```
[32] n_sen_chunk_3 = n_sentenize(text3)
     n_sen_chunk_3
```

```
[['Содержит',
  'ли',
  'это',
  'излучение',
  'информацию',
  'об',
  'их',
  'внутренней',
  'структуре',
  ',',
  'как',
  'это',
  'предполагает',
  'дуальность',
  'тяготение-калибровочная',
  'инвариантность',
  ',',
  'или',
  'нет',
  ',',
  'как',
  'следует',
  'из',
  'оригинального',
  'расчёта',
  'Хокинга',
  '?']]
```

Частеречная разметка

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

```
# Файл необходимо скачать по ссылке https://github.com/natasha/navec#downloads
navec = Navec.load('navec_news_v1_1B_250K_300d_100q.tar')
```

```
# Файл необходимо скачать по ссылке https://github.com/natasha/slovnet#downloads
n_morph = Morph.load('slovnet_morph_news_v1.tar', batch_size=4)
```

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

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

```
n_text1_markup = list(_ for _ in n_morph.map(n_sen_chunk_1))
[print_pos(x) for x in n_text1_markup]
```

```
Адаптивная - ADJ|Case=Nom|Degree=Pos|Gender=Fem|Number=Sing
вёрстка - NOUN|Animacy=Inan|Case=Nom|Gender=Fem|Number=Sing
меняет - VERB|Aspect=Imp|Mood=Ind|Number=Sing|Person=3|Tense=Pres|VerbForm=Fin|Voice=Act
дизайн - NOUN|Animacy=Inan|Case=Acc|Gender=Masc|Number=Sing
страницы - NOUN|Animacy=Inan|Case=Gen|Gender=Fem|Number=Sing
в - ADP
зависимости - NOUN|Animacy=Inan|Case=Loc|Gender=Fem|Number=Sing
от - ADP
поведения - NOUN|Animacy=Inan|Case=Gen|Gender=Neut|Number=Sing
пользователя - NOUN|Animacy=Anim|Case=Gen|Gender=Masc|Number=Sing
, - PUNCT
платформы - NOUN|Animacy=Inan|Case=Gen|Gender=Fem|Number=Sing
, - PUNCT
размера - NOUN|Animacy=Inan|Case=Gen|Gender=Masc|Number=Sing
экрана - NOUN|Animacy=Inan|Case=Gen|Gender=Masc|Number=Sing
и - CCONJ
ориентации - NOUN|Animacy=Inan|Case=Gen|Gender=Fem|Number=Sing
девайса - NOUN|Animacy=Inan|Case=Gen|Gender=Masc|Number=Sing
и - CCONJ
является - VERB|Aspect=Imp|Mood=Ind|Number=Sing|Person=3|Tense=Pres|VerbForm=Fin|Voice=Mid
неотъемлемой - ADJ|Case=Ins|Degree=Pos|Gender=Fem|Number=Sing
частью - NOUN|Animacy=Inan|Case=Ins|Gender=Fem|Number=Sing
современной - ADJ|Case=Gen|Degree=Pos|Gender=Fem|Number=Sing
веб-разработки - NOUN|Animacy=Inan|Case=Gen|Gender=Fem|Number=Sing
. - PUNCT
[None]
```

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

```
Покупатели – NOUN|Animacy=Anim|Case=Nom|Gender=Masc|Number=Plur
могут – VERB|Aspect=Imp|Mood=Ind|Number=Plur|Person=3|Tense=Pres|VerbForm=Fin|Voice=Act
получать – VERB|Aspect=Imp|VerbForm=Inf|Voice=Act
заказы – NOUN|Animacy=Inan|Case=Acc|Gender=Masc|Number=Plur
в – ADP
пунктах – NOUN|Animacy=Inan|Case=Loc|Gender=Masc|Number=Plur
выдачи – NOUN|Animacy=Inan|Case=Gen|Gender=Fem|Number=Sing
заказов – NOUN|Animacy=Inan|Case=Gen|Gender=Masc|Number=Plur
, – PUNCT
в – ADP
почтоматах – NOUN|Animacy=Inan|Case=Loc|Gender=Masc|Number=Sing
( – PUNCT
преимуществом – NOUN|Animacy=Inan|Case=Ins|Gender=Neut|Number=Sing
этого – DET|Case=Gen|Gender=Masc|Number=Sing
варианта – NOUN|Animacy=Inan|Case=Gen|Gender=Masc|Number=Sing
является – VERB|Aspect=Imp|Mood=Ind|Number=Sing|Person=3|Tense=Pres|VerbForm=Fin|Voice=Mid
экономия – NOUN|Animacy=Inan|Case=Nom|Gender=Fem|Number=Sing
на – ADP
оплате – NOUN|Animacy=Inan|Case=Loc|Gender=Fem|Number=Sing
труда – NOUN|Animacy=Inan|Case=Gen|Gender=Masc|Number=Sing
) – PUNCT
или – CCONJ
заказы – NOUN|Animacy=Inan|Case=Nom|Gender=Masc|Number=Plur
могут – VERB|Aspect=Imp|Mood=Ind|Number=Plur|Person=3|Tense=Pres|VerbForm=Fin|Voice=Act
доставляться – VERB|Aspect=Imp|VerbForm=Inf|Voice=Pass
им – PRON|Case=Dat|Number=Plur|Person=3
непосредственно – ADV|Degree=Pos
до – ADP
дома – NOUN|Animacy=Inan|Case=Gen|Gender=Masc|Number=Sing
( – PUNCT
или – CCONJ
места – NOUN|Animacy=Inan|Case=Gen|Gender=Neut|Number=Sing
работы – NOUN|Animacy=Inan|Case=Gen|Gender=Fem|Number=Sing
) – PUNCT
курьерами – NOUN|Animacy=Anim|Case=Ins|Gender=Masc|Number=Plur
. – PUNCT
[None]
```

```
n_text3_markup = list(n_morph.map(n_sen_chunk_3))
[print_pos(x) for x in n_text3_markup]
```

```
Содержит – VERB|Aspect=Imp|Mood=Ind|Number=Sing|Person=3|Tense=Pres|VerbForm=Fin|Voice=Act
ли – PART
это – DET|Case=Nom|Gender=Neut|Number=Sing
излучение – NOUN|Animacy=Inan|Case=Nom|Gender=Neut|Number=Sing
информацию – NOUN|Animacy=Inan|Case=Acc|Gender=Fem|Number=Sing
об – ADP
их – DET
внутренней – ADJ|Case=Loc|Degree=Pos|Gender=Fem|Number=Sing
структуре – NOUN|Animacy=Inan|Case=Loc|Gender=Fem|Number=Sing
, – PUNCT
как – SCONJ
это – PRON|Animacy=Inan|Case=Nom|Gender=Neut|Number=Sing
предполагает – VERB|Aspect=Imp|Mood=Ind|Number=Sing|Person=3|Tense=Pres|VerbForm=Fin|Voice=Act
дуальность – ADJ|Case=Ins|Degree=Pos|Gender=Masc|Number=Sing
тяготение-калибровочная – ADJ|Case=Ins|Degree=Pos|Gender=Masc|Number=Sing
инвариантность – NOUN|Animacy=Inan|Case=Gen|Gender=Masc|Number=Sing
, – PUNCT
или – CCONJ
нет – PART
, – PUNCT
как – SCONJ
следует – VERB|Aspect=Imp|Mood=Ind|Number=Sing|Person=3|Tense=Pres|VerbForm=Fin|Voice=Act
из – ADP
оригинального – ADJ|Case=Gen|Degree=Pos|Gender=Neut|Number=Sing
расчёта – NOUN|Animacy=Inan|Case=Gen|Gender=Neut|Number=Sing
Хокинга – PROPN|Animacy=Anim|Case=Gen|Gender=Masc|Number=Sing
? – PUNCT
[None]
```

Лемматизация

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

```
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
```

```
n_doc1 = n_lemmatize(text1)
{_.text: _.lemma for _ in n_doc1.tokens}
```

```
{'Адаптивная': 'адаптивный',
 'вёрстка': 'верстка',
 'меняет': 'менять',
 'дизайн': 'дизайн',
 'страницы': 'страница',
 'в': 'в',
 'зависимости': 'зависимость',
 'от': 'от',
 'поведения': 'поведение',
 'пользователя': 'пользователь',
 ',': ',',
 'платформы': 'платформа',
 'размера': 'размер',
 'экрана': 'экран',
 'и': 'и',
 'ориентации': 'ориентация',
 'девайса': 'девайс',
 'является': 'являться',
 'неотъемлемой': 'неотъемлемый',
 'частью': 'часть',
 'современной': 'современный',
 'веб-разработки': 'веб-разработка',
 '.': '.'}
```

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

```
{'Покупатели': 'покупатель',
 'могут': 'мочь',
 'получать': 'получать',
 'заказы': 'заказ',
 'в': 'в',
 'пунктах': 'пункт',
 'выдачи': 'выдача',
 'заказов': 'заказ',
 ',': ',',
 'почтоматах': 'почтомат',
 '(': '(',
 'преимуществом': 'преимущество',
 'этого': 'этот',
 'варианта': 'вариант',
 'является': 'являться',
 'экономия': 'экономия',
 'на': 'на',
 'оплате': 'оплата',
 'труда': 'труд',
 ')': ')',
 'или': 'или',
 'доставляться': 'доставляться',
 'им': 'они',
 'непосредственно': 'непосредственно',
 'до': 'до',
 'дома': 'дом',
 'места': 'место',
 'работы': 'работа',
 'курьерами': 'курьер',
 '.': '.'}
```

```
n_doc3 = n_lemmatize(text3)
{_.text: _.lemma for _ in n_doc3.tokens}
```

```
{'Содержит': 'содержать',
 'ли': 'ли',
 'это': 'это',
 'излучение': 'излучение',
 'информацию': 'информация',
 'об': 'о',
 'их': 'их',
 'внутренней': 'внутренний',
 'структуре': 'структура',
 ',': ',',
 'как': 'как',
 'предполагает': 'предполагать',
 'дуальность': 'дуальность',
 'тяготение-калибровочная': 'тяготение-калибровочный',
 'инвариантность': 'инвариантность',
 'или': 'или',
 'нет': 'нет',
 'следует': 'следовать',
 'из': 'из',
 'оригинального': 'оригинальный',
 'расчёта': 'расчет',
 'Хокинга': 'хокинг',
 '?': '?'}
```

Выделение (распознавание именованных сущностей)

```
from slovnet import NER
from ipymarkup import show_span_ascii_markup as show_markup
```

```
ner = NER.load('slovnet_ner_news_v1.tar')
ner_res = ner.navec(navec)
markup_ner3 = ner(text3)
```

```
markup_ner3
```

```
SpanMarkup(
    text='Содержит ли это излучение информацию об их внутренней структуре, как это предполагает дуальность тяготение-калибровочная инвариантность, или нет, как следует из оригинального расчёта Хокинга?',
    spans=[Span(
        start=183,
        stop=190,
        type='PER'
    )]
)
```

```
show_markup(markup_ner3.text, markup_ner3.spans)
```

```
Содержит ли это излучение информацию об их внутренней структуре, как
```

это предполагает дуальность тяготение-калибровочная инвариантность,

или нет, как следует из оригинального расчёта Хокинга?

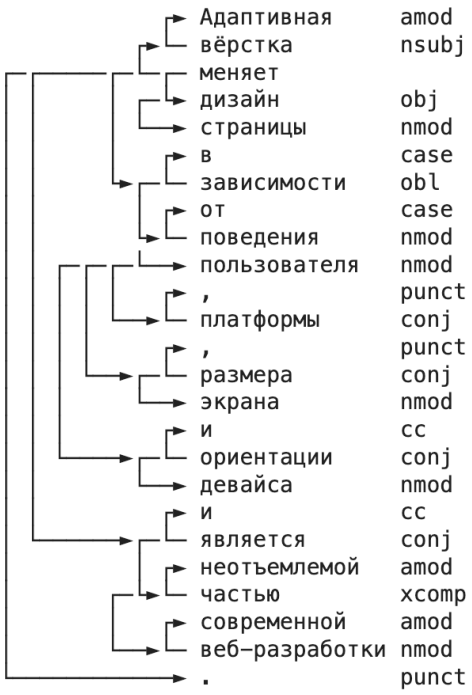
PER———

Разбор предложения

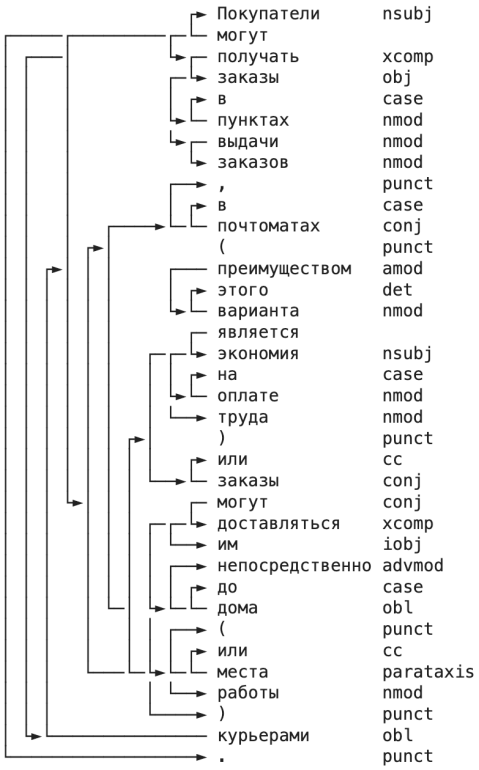
```
from natasha import NewsSyntaxParser
```

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

```
n_doc1.parse_syntax(syntax_parser)
n_doc1.sents[0].syntax.print()
```




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



```
n_doc3.parse_syntax(syntax_parser)
n_doc3.sents[0].syntax.print()
```