МОСКОВСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
им. Н.Э. Баумана

Факультет «Информатика и системы управления»
Кафедра «Систем обработки информации и управления»

# ОТЧЕТ

**Лабораторная работа №__8__**
по дисциплине «Предобработка текста»

Тема: «Обучение на основе глубоких Q-сетей»

ИСПОЛНИТЕЛЬ: ___Алексеев_А_С__
ФИО
группа ИУ5-25 _____
подпись
"__"_____2024 г.

ПРЕПОДАВАТЕЛЬ: ___Гапанюк_Ю_Е__
ФИО
_____
подпись
"__"_____2024 г.

Москва - 2024

## Задание

Для произвольного предложения или текста решите следующие задачи:

1. Токенизация.
2. Частеречная разметка.
3. Лемматизация.
4. Выделение (распознавание) именованных сущностей.
5. Разбор предложения.

Текст

```
In [1]:text1 = 'Предобработка текста — это первый и один из наиболее важных этапов в обработке естественного языка (NLP) с использованием нейр
       text2 = 'Шла Саша по шоссе и сосала сушку.'
       text3 = 'Санкт-Петербург — один из крупнейших промышленных центров России, лидер по производству машиностроительной продукции, созд
```

Токенизация

```
In [2]:import nltk
       from nltk.tokenize import punkt
       nltk.download('punkt')
```

```
[nltk_data] Downloading package punkt to
[nltk_data]     C:\Users\aleka\AppData\Roaming\nltk_data...
[nltk_data]   Package punkt is already up-to-date!
Out[2]:True
In [3]:from nltk import tokenize
       dir(tokenize)[:18]
```

```
Out[3]:['BlanklineTokenizer',
         'LegalitySyllableTokenizer',
         'LineTokenizer',
         'MWETokenizer',
         'NLTKWordTokenizer',
         'PunktSentenceTokenizer',
         'RegexpTokenizer',
         'ReppTokenizer',
         'SExprTokenizer',
         'SpaceTokenizer',
         'StanfordSegmenter',
         'SyllableTokenizer',
         'TabTokenizer',
         'TextTilingTokenizer',
         'ToktokTokenizer',
         'TreebankWordDetokenizer',
         'TreebankWordTokenizer',
         'TweetTokenizer']
In [4]:nltk_tk_1 = nltk.WordPunctTokenizer()
       nltk_tk_1.tokenize(text1)
```

Out[4]:['Предобработка',
'текста',
'—',
'это',
'первый',
'и',
'один',
'из',
'наиболее',
'важных',
'этапов',
'в',
'обработке',
'естественного',
'языка',
'(',
'NLP',
')',
'с',
'использованием',
'нейронных',
'сетей',
'.',
'Этот',
'процесс',
'включает',
'в',
'себя',
'серию',
'операций',
',',
'предназначенных',
'для',
'преобразования',
'исходного',
'текста',
'в',
'формат',
',',
'который',
'может',
'быть',
'эффективно',
'обработан',
'нейронными',
'сетями',
'.']

In [5]:*# Токенизация по предложениям*
```
nltk_tk_sents = nltk.tokenize.sent_tokenize(text1)
print(len(nltk_tk_sents))
nltk_tk_sents
```

2

Out[5]:['Предобработка текста — это первый и один из наиболее важных этапов в обработке естественного языка (NLP) с использованием нейронн
ых сетей.',
'Этот процесс включает в себя серию операций, предназначенных для преобразования исходного текста в формат, который может быть эф
фективно обработан нейронными сетями.']

In [6]:
```
from razdel import tokenize, sentenize
```

In [7]:
```
n_tok_text1 = list(tokenize(text1))
n_tok_text1
```

Out[7]:[Substring(0, 13, 'Предобработка'),
 Substring(14, 20, 'текста'),
 Substring(21, 22, '—'),
 Substring(23, 26, 'это'),
 Substring(27, 33, 'первый'),
 Substring(34, 35, 'и'),
 Substring(36, 40, 'один'),
 Substring(41, 43, 'из'),
 Substring(44, 52, 'наиболее'),
 Substring(53, 59, 'важных'),
 Substring(60, 66, 'этапов'),
 Substring(67, 68, 'в'),
 Substring(69, 78, 'обработке'),
 Substring(79, 92, 'естественного'),
 Substring(93, 98, 'языка'),
 Substring(99, 100, '('),
 Substring(100, 103, 'NLP'),
 Substring(103, 104, ')'),
 Substring(105, 106, 'с'),
 Substring(107, 121, 'использованием'),
 Substring(122, 131, 'нейронных'),
 Substring(132, 137, 'сетей'),
 Substring(137, 138, '.'),
 Substring(139, 143, 'Этот'),
 Substring(144, 151, 'процесс'),
 Substring(152, 160, 'включает'),
 Substring(161, 162, 'в'),
 Substring(163, 167, 'себя'),
 Substring(168, 173, 'серию'),
 Substring(174, 182, 'операций'),
 Substring(182, 183, ','),
 Substring(184, 199, 'предназначенных'),
 Substring(200, 203, 'для'),
 Substring(204, 218, 'преобразования'),
 Substring(219, 228, 'исходного'),
 Substring(229, 235, 'текста'),
 Substring(236, 237, 'в'),
 Substring(238, 244, 'формат'),
 Substring(244, 245, ','),
 Substring(246, 253, 'который'),
 Substring(254, 259, 'может'),
 Substring(260, 264, 'быть'),
 Substring(265, 275, 'эффективно'),
 Substring(276, 285, 'обработан'),
 Substring(286, 296, 'нейронными'),
 Substring(297, 303, 'сетями'),
 Substring(303, 304, '.')]

In [8]:
```
[_.text for _ in n_tok_text1]
```

Out[8]:['Предобработка',
'текста',
'—',
'это',
'первый',
'и',
'один',
'из',
'наиболее',
'важных',
'этапов',
'в',
'обработке',
'естественного',
'языка',
'(',
'NLP',
')',
'с',
'использованием',
'нейронных',
'сетей',
'.',
'Этот',
'процесс',
'включает',
'в',
'себя',
'серию',
'операций',
',',
'предназначенных',
'для',
'преобразования',
'исходного',
'текста',
'в',
'формат',
',',
'который',
'может',
'быть',
'эффективно',
'обработан',
'нейронными',
'сетями',
'.']

In [9]:
```
n_sen_text1 = list(sentenize(text1))
n_sen_text1
```

Out[9]:
```
[Substring(0,
           138,
           'Предобработка текста — это первый и один из наиболее важных этапов в обработке естественного языка (NLP) с использованием ней
ронных сетей.'),
 Substring(139,
           304,
           'Этот процесс включает в себя серию операций, предназначенных для преобразования исходного текста в формат, который может бы
ть эффективно обработан нейронными сетями.')]
```

In [10]:
```
[_.text for _ in n_sen_text1], len([_.text for _ in n_sen_text1])
```

Out[10]:
```
(['Предобработка текста — это первый и один из наиболее важных этапов в обработке естественного языка (NLP) с использованием нейрон
ных сетей.',
  'Этот процесс включает в себя серию операций, предназначенных для преобразования исходного текста в формат, который может быть э
ффективно обработан нейронными сетями.'],
 2)
```

In [11]:
```
# 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 [12]:
```
n_sen_chunk_1 = n_sentenize(text1)
n_sen_chunk_1
```

```
Out[12]:[['Предобработка',
          'текста',
          '—',
          'это',
          'первый',
          'и',
          'один',
          'из',
          'наиболее',
          'важных',
          'этапов',
          'в',
          'обработке',
          'естественного',
          'языка',
          '(',
          'NLP',
          ')',
          'с',
          'использованием',
          'нейронных',
          'сетей',
          '.'],
         ['Этот',
          'процесс',
          'включает',
          'в',
          'себя',
          'серию',
          'операций',
          ',',
          'предназначенных',
          'для',
          'преобразования',
          'исходного',
          'текста',
          'в',
          'формат',
          ',',
          'который',
          'может',
          'быть',
          'эффективно',
          'обработан',
          'нейронными',
          'сетями',
          '.']]
In [13]:n_sen_chunk_2 = n_sentenize(text2)
        n_sen_chunk_2

Out[13]:[['Шла', 'Саша', 'по', 'шоссе', 'и', 'сосала', 'сушку', '.']]
In [14]:n_sen_chunk_3 = n_sentenize(text3)
        n_sen_chunk_3
```

Out[14]:[['Санкт-Петербург',
'—',
'один',
'из',
'крупнейших',
'промышленных',
'центров',
'России',
',',
'лидер',
'по',
'производству',
'машиностроительной',
'продукции',
',',
'создаёт',
'11',
'%',
'от',
'общего',
'объёма',
'продукции',
'машиностроения',
'по',
'стране',
'.']]

Частеречная разметка

In [15]:
```
from navec import Navec
from slovnet import Morph
```

In [16]:
```
# Файл необходимо скачать по ссылке https://github.com/natasha/navec#downloads
navec = Navec.load('navec_news_v1_1B_250K_300d_100q.tar')
```

In [17]:
```
# Файл необходимо скачать по ссылке https://github.com/natasha/slovnet#downloads
n_morph = Morph.load('slovnet_morph_news_v1.tar', batch_size=4)
```

In [18]:
```
morph_res = n_morph.navec(navec)
```

In [19]:
```
def print_pos(markup):
    for token in markup.tokens:
        print('{} - {}'.format(token.text, token.tag))
```

In [20]:
```
n_text1_markup = list(_ for _ in n_morph.map(n_sen_chunk_1))
[print_pos(x) for x in n_text1_markup]
```

Предобработка - ADJ|Case=Gen|Degree=Pos|Gender=Masc|Number=Sing
текста - NOUN|Animacy=Inan|Case=Gen|Gender=Masc|Number=Sing
— - PUNCT
это - PART
первый - ADJ|Case=Nom|Degree=Pos|Gender=Masc|Number=Sing
и - CCONJ
один - NUM|Case=Nom|Gender=Masc|Number=Sing
из - ADP
наиболее - ADV|Degree=Pos
важных - ADJ|Case=Gen|Degree=Pos|Number=Plur
этапов - NOUN|Animacy=Inan|Case=Gen|Gender=Masc|Number=Plur
в - ADP
обработке - NOUN|Animacy=Inan|Case=Loc|Gender=Fem|Number=Sing
естественного - ADJ|Case=Gen|Degree=Pos|Gender=Masc|Number=Sing
языка - NOUN|Animacy=Inan|Case=Gen|Gender=Masc|Number=Sing
( - PUNCT
NLP - PROPN|Foreign=Yes
) - PUNCT
с - ADP
использованием - NOUN|Animacy=Inan|Case=Ins|Gender=Neut|Number=Sing
нейронных - ADJ|Case=Gen|Degree=Pos|Number=Plur
сетей - NOUN|Animacy=Inan|Case=Gen|Gender=Fem|Number=Plur
. - PUNCT
Этот - DET|Case=Nom|Gender=Masc|Number=Sing
процесс - NOUN|Animacy=Inan|Case=Nom|Gender=Masc|Number=Sing
включает - VERB|Aspect=Imp|Mood=Ind|Number=Sing|Person=3|Tense=Pres|VerbForm=Fin|Voice=Act
в - ADP
себя - PRON|Case=Acc
серию - NOUN|Animacy=Inan|Case=Acc|Gender=Fem|Number=Sing
операций - NOUN|Animacy=Inan|Case=Gen|Gender=Fem|Number=Plur
, - PUNCT
предназначенных - VERB|Aspect=Perf|Case=Gen|Number=Plur|Tense=Past|VerbForm=Part|Voice=Pass
для - ADP
преобразования - NOUN|Animacy=Inan|Case=Gen|Gender=Neut|Number=Sing
исходного - ADJ|Case=Gen|Degree=Pos|Gender=Masc|Number=Sing
текста - NOUN|Animacy=Inan|Case=Gen|Gender=Masc|Number=Sing
в - ADP
формат - NOUN|Animacy=Inan|Case=Acc|Gender=Masc|Number=Sing
, - PUNCT
который - PRON|Case=Nom|Gender=Masc|Number=Sing
может - VERB|Aspect=Imp|Mood=Ind|Number=Sing|Person=3|Tense=Pres|VerbForm=Fin|Voice=Act
быть - AUX|Aspect=Imp|VerbForm=Inf|Voice=Act
эффективно - ADV|Degree=Pos
обработан - VERB|Aspect=Perf|Gender=Masc|Number=Sing|Tense=Past|Variant=Short|VerbForm=Part|Voice=Pass
нейронными - ADJ|Case=Ins|Degree=Pos|Number=Plur
сетями - NOUN|Animacy=Inan|Case=Ins|Gender=Fem|Number=Plur
. - PUNCT
Out[20]:[None, None]
In [21]:n_text2_markup = list(n_morph.map(n_sen_chunk_2))
[print_pos(x) for x in n_text2_markup]
Шла - VERB|Aspect=Imp|Gender=Fem|Mood=Ind|Number=Sing|Tense=Past|VerbForm=Fin|Voice=Act
Саша - PROPN|Animacy=Anim|Case=Nom|Gender=Masc|Number=Sing
по - ADP
шоссе - NOUN|Animacy=Inan|Case=Dat|Gender=Neut|Number=Sing
и - CCONJ
сосала - ADJ|Case=Acc|Degree=Pos|Gender=Fem|Number=Sing
сушку - NOUN|Animacy=Inan|Case=Acc|Gender=Fem|Number=Sing
. - PUNCT
Out[21]:[None]
In [22]:n_text3_markup = list(n_morph.map(n_sen_chunk_3))
[print_pos(x) for x in n_text3_markup]

Санкт-Петербург - PROPN|Animacy=Inan|Case=Nom|Gender=Masc|Number=Sing
— - PUNCT
один - NUM|Case=Nom|Gender=Masc|Number=Sing
из - ADP
крупнейших - ADJ|Case=Gen|Degree=Sup|Number=Plur
промышленных - ADJ|Case=Gen|Degree=Pos|Number=Plur
центров - NOUN|Animacy=Inan|Case=Gen|Gender=Masc|Number=Plur
России - PROPN|Animacy=Inan|Case=Gen|Gender=Fem|Number=Sing
, - PUNCT
лидер - NOUN|Animacy=Anim|Case=Nom|Gender=Masc|Number=Sing
по - ADP
производству - NOUN|Animacy=Inan|Case=Dat|Gender=Neut|Number=Sing
машиностроительной - ADJ|Case=Gen|Degree=Pos|Gender=Fem|Number=Sing
продукции - NOUN|Animacy=Inan|Case=Gen|Gender=Fem|Number=Sing
, - PUNCT
создаёт - VERB|Aspect=Imp|Mood=Ind|Number=Sing|Person=3|Tense=Pres|VerbForm=Fin|Voice=Act
11 - NUM
% - SYM
от - ADP
общего - ADJ|Case=Gen|Degree=Pos|Gender=Masc|Number=Sing
объёма - NOUN|Animacy=Inan|Case=Gen|Gender=Masc|Number=Sing
продукции - NOUN|Animacy=Inan|Case=Gen|Gender=Fem|Number=Sing
машиностроения - NOUN|Animacy=Inan|Case=Gen|Gender=Neut|Number=Sing
по - ADP
стране - NOUN|Animacy=Inan|Case=Dat|Gender=Fem|Number=Sing
. - PUNCT
Out[22]:[None]
Лемматизация

In [23]:
```
from natasha import Doc, Segmenter, NewsEmbedding, NewsMorphTagger, MorphVocab
```
In [24]:
```
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 [25]:
```
n_doc1 = n_lemmatize(text1)
{_.text: _.lemma for _ in n_doc1.tokens}
```

Out[25]:{'Предобработка': 'предобработка',
'текста': 'текст',
'—': '—',
'это': 'это',
'первый': 'первый',
'и': 'и',
'один': 'один',
'из': 'из',
'наиболее': 'наиболее',
'важных': 'важный',
'этапов': 'этап',
'в': 'в',
'обработке': 'обработка',
'естественного': 'естественный',
'языка': 'язык',
'(': '(',
'NLP': 'nlp',
')': ')',
'с': 'с',
'использованием': 'использование',
'нейронных': 'нейронный',
'сетей': 'сеть',
'.': '.',
'Этот': 'этот',
'процесс': 'процесс',
'включает': 'включать',
'себя': 'себя',
'серию': 'серия',
'операций': 'операция',
',': ',',
'предназначенных': 'предназначить',
'для': 'для',
'преобразования': 'преобразование',
'исходного': 'исходный',
'формат': 'формат',
'который': 'который',
'может': 'мочь',
'быть': 'быть',
'эффективно': 'эффективно',
'обработан': 'обработать',
'нейронными': 'нейронный',
'сетями': 'сеть'}

In [26]:
```
n_doc2 = n_lemmatize(text2)
{_.text: _.lemma for _ in n_doc2.tokens}
```

Out[26]:{'Шла': 'идти',
'Саша': 'саша',
'по': 'по',
'шоссе': 'шоссе',
'и': 'и',
'сосала': 'сосать',
'сушку': 'сушка',
'.': '.'}

In [27]:
```
n_doc3 = n_lemmatize(text3)
{_.text: _.lemma for _ in n_doc3.tokens}
```

Out[27]:{'Санкт-Петербург': 'санкт-петербург',
'—': '—',
'один': 'один',
'из': 'из',
'крупнейших': 'крупный',
'промышленных': 'промышленный',
'центров': 'центр',
'России': 'россия',
',': ',',
'лидер': 'лидер',
'по': 'по',
'производству': 'производство',
'машиностроительной': 'машиностроительный',
'продукции': 'продукция',
'создаёт': 'создавать',
'11': '11',
'%': '%',
'от': 'от',
'общего': 'общий',
'объёма': 'объем',
'машиностроения': 'машиностроение',
'стране': 'страна',
'.': '.'}

Выделение (распознавание) именованных сущностей

```
In [28]:from slovnet import NER
        from ipymarkup import show_span_ascii_markup as show_markup
In [29]:ner = NER.load('slovnet_ner_news_v1.tar')
        ner_res = ner.navec(navec)
        markup_ner3 = ner(text3)
In [30]:markup_ner3
Out[30]:SpanMarkup(
            text='Санкт-Петербург — один из крупнейших промышленных центров России, лидер по производству машиностроительной продукции, с
         оздаёт 11% от общего объёма продукции машиностроения по стране.',
            spans=[Span(
                start=0,
                stop=15,
                type='LOC'
            ),
            Span(
                start=58,
                stop=64,
                type='LOC'
            )]
        )
In [31]:show_markup(markup_ner3.text, markup_ner3.spans)
Санкт-Петербург — один из крупнейших промышленных центров России,
LOC————————————                                    LOC———
лидер по производству машиностроительной продукции, создаёт 11% от
общего объёма продукции машиностроения по стране.
```

Разбор предложения

```
In [32]:from natasha import NewsSyntaxParser
In [33]:emb = NewsEmbedding()
        syntax_parser = NewsSyntaxParser(emb)
In [34]:n_doc1.parse_syntax(syntax_parser)
        n_doc1.sents[0].syntax.print()
          ┌─► Предобработка  amod
        ┌─└── текста
        └────► —             punct
    ┌──────────────────► это           expl
    │  ┌─────► ┌──────── первый        parataxis
    │  │    │  ┌─► и             cc
    │  │    └─►└── один           conj
    │  │  ┌──────────► из            case
    │  │  │    ┌─► наиболее       advmod
    │  │  │  ┌─► └── важных        amod
┌───────── └──└──└──└─────── этапов        nmod
│      │  │    ┌─► в             case
│    ┌─└─►└─────── └── обработке      nmod
│    │       ┌─► естественного  amod
│  ┌─└──────────────► └── языка         nmod
│  │         ┌─► (             punct
│  └──────────────► ┌─└── NLP           appos
│         └─────► )             punct
│           ┌─► с             case
└─────────────────► ┌─└── использованием nmod
        │  ┌─► нейронных      amod
        └─►└── сетей          nmod
            .
In [35]:n_doc1.parse_syntax(syntax_parser)
        n_doc1.sents[1].syntax.print()
```

```
        ┌► Этот          det
      ┌► └── процесс     nsubj
┌──┌──┌──┌──└──────── включает
│  │  │  │   ┌► в          case
│  │  │  └──────► └── себя       obl
│  │  └──────────► ┌── серию     obj
│  │     ┌── └► операций    nmod
│  │     │  ┌► ,           punct
│  │   ┌── └► └── предназначенных acl
│  │   │   ┌► для         case
│  │  ┌── └──────► └── преобразования obl
│  │  │      ┌► исходного    amod
│  │  └──────────► └── текста      nmod
│  │        ┌► в          case
│  └──────► ┌──────── └── формат      obl
│     │  ┌──────► ,           punct
│     │  │  ┌► который      nsubj:pass
│   ┌── └► └── └── может       acl:relcl
│   │   ┌──────► быть         aux:pass
│   │   │  ┌► эффективно   advmod
│   └► ┌── └── └── обработан    xcomp
│     │   ┌► нейронными    amod
│     └──────► └── сетями      obl
└──────────────────────► .          punct
```

In [36]:
```
n_doc2.parse_syntax(syntax_parser)
n_doc2.sents[0].syntax.print()
```

```
   Шла
   Саша
  ┌► по    case
  └── шоссе
┌──────► и     cc
│  ┌──┌► сосала amod
└── └► └── сушку  obj
   .
```

In [37]:
```
n_doc3.parse_syntax(syntax_parser)
n_doc3.sents[0].syntax.print()
```

```
     ┌──────► Санкт-Петербург    nsubj
     │  ┌► —              punct
┌──┌──┌──────── └── └── один
│  │  │  ┌──────────► из             case
│  │  │  │  ┌──────► крупнейших      amod
│  │  │  │  │  ┌► промышленных    amod
│  │  └► └── └── └── центров        nmod
│  │     └──────► России          nmod
│  │        ┌► ,              punct
│  └──────────► ┌── └── лидер           conj
│     │  ┌► по             case
│   ┌── └► └── производству    nmod
│   │   ┌► машиностроительной amod
│   └──────► └── продукции      nmod
│       ┌► ,              punct
│   ┌── ┌── └── создаёт
│   │  │  ┌► 11             nummod
│   │  └► └── %              obj
│   │  ┌──────► от             case
│   │  │  ┌► общего         amod
│   └► └── └── объёма         obl
│   ┌── └► ┌── продукции      nmod
│   │    └► машиностроения  nmod
│   │   ┌► по             case
│   └──────► └── стране         nmod
└──────────────────────► .             punct
```

In [ ]: