Практическое занятие №2.3.3. Библиотека natasha. Сегментация на токены и предложения, морфологический и синтаксический анализ, лемматизация, извлечение именованных сущностей

На предыдущем занятии мы рассматривали библиотеку NLTK для символьной и статистической обработки естественного языка. NLTK более приспособлена для обработки текстов на английском языке. Сегодня мы познакомимся с библиотекой natasha, предназначенной для обработки естественного русского языка и разработанной в лаборатории анализа данных Александра Кукушкина. Библиотека позволяет решать все базовые задачи по обработке текстов.

Данное практическое занятие возникло из статьи Алеси Мараховской, которая поведала о расследовании, связанном с поиском штатных понятых, которые помогают полицейским фальсифицировать дела за наркотики. Штатные понятые – это люди, которые, как правило, знакомы с полицейскими или находятся от них в зависимости, потому что они уже сами судимы или накануне задержаны, и над ними висит угроза нового срока. Вместо честных показаний, такие понятые могут просто подтверждать всё, что скажут сотрудники полиции. И не по одному разу.

Как найти таких понятых? Они обычно выступают в суде или там оглашаются их показания. Поэтому таких понятых можно поискать в приговорах судов. В рамках расследования были скачаны с сайта Мосгорсуда документы по всем наркотическим статьям. А дальше нужно было извлечь из текстов все имена и найти те, что встречаются не только в одном деле, а в двух и более.

На практическом занятии мы разберем ту часть работы, которая связана с извлечением имен из текста. Для этого будем использовать библиотеку natasha, которая решает все базовые задачи обработки естественного русского языка: сегментация на токены и предложения, морфологический и синтаксический анализ, лемматизация, извлечение именованных сущностей.

Давайте на примере одного приговора посмотрим, как это работает. Текст приговора (Приговор.txt) вы должны скачать из данного раздела на открытом портале.

Извлечение имен и фамилий из текстов на русском языке с помощью библиотеки natasha

Перед началом работы мы предлагаем вам запустить программный код, который позволит загрузить ваш файл с текстом приговора.

```
# Загрузить любой файл с компьютера в google.colab
from google.colab import files
```

```
uploaded = files.upload()

for fn in uploaded.keys():
    print('User uploaded file "{name}" with length {length}
bytes'.format(
      name=fn, length=len(uploaded[fn])))
```

[]: • **Приговор.txt**(text/plain) - 28053 bytes, last modified: 14.11.2021 - 100% done

```
Saving Приговор.txt to Приговор.txt
User uploaded file "Приговор.txt" with length 28053 bytes
```

Теперь экспортируем библиотеку и пропишем все необходимы строки для дальнейшей обработки текста.

```
!pip install natasha
import natasha as nt

segmenter = nt.Segmenter()
morph_vocab = nt.MorphVocab()

emb = nt.NewsEmbedding()
morph_tagger = nt.NewsMorphTagger(emb)
syntax_parser = nt.NewsSyntaxParser(emb)
ner_tagger = nt.NewsNERTagger(emb)

names_extractor = nt.NamesExtractor(morph_vocab)
dates_extractor = nt.DatesExtractor(morph_vocab)
money_extractor = nt.MoneyExtractor(morph_vocab)
addr_extractor = nt.AddrExtractor(morph_vocab)

file = open('Приговор.txt')
text = file.read()
print(text)
```

[]:

```
ПРИГОВОР

ИМЕНЕМ  РОССИЙСКОЙ ФЕДЕРАЦИИ
01 декабря 2016 года
Москва
Преображенский районный суд Москвы в составе судьи Духновской З.А., при
секретаре Горшковой А.А., с участием государственных обвинителей Давыдовой
Д.Е., Дудукова М.Ю., Чубенко А.А., подсудимого **********а А.А., его
защитников – адвокатов Попова А.В.,  Колокольниковой Е.В., рассмотрев в
открытом судебном заседании уголовное дело по обвинению:
********** ****** **********, ** ** **** года рождения, уроженца г.
Москвы, гражданина РФ, зарегистрированного по адресу: **********, со
средне-специальным образованием, холостого, не работающего, не судимого,  в
совершении преступления, предусмотренного ч. 3 ст. 30, п.«г» ч. 4 ст. 228-1 УК
РФ,
УСТАНОВИЛ:
********** ****** ******** совершил покушение на незаконный сбыт
наркотических средств, то есть умышленные действия, непосредственно
направленные на незаконный сбыт наркотических средств, совершенные группой лиц
```

```
по предварительному сговору, в крупном размере, при этом преступление не было
доведено до конца по не зависящим от него обстоятельствам, а именно:
Так ************ А.А., с целью быстрого и незаконного материального
обогащения, из корыстных побуждений, имея преступный умысел, направленный на
незаконный сбыт наркотических средств группой лиц по предварительному сговору,
в крупном размере, осознавая общественную опасность последствий своих действий
и желая их наступления, в нарушение ст. ст. 5, 8, 14, 20 и
```

В этом приговоре довольно много разных имен, на которых можно проверить, как работает библиотека. Для начала передадим текст нашей библиотеке. Назовем этот объект переменной doc и совершим с ней все возможные действия, на которые способна библиотека.

```
doc = nt.Doc(text)
doc.segment(segmenter)
doc.tag_morph(morph_tagger)
doc.parse_syntax(syntax_parser)
doc.tag_ner(ner_tagger)
```

Проверим, как библиотека справляет с сегментацией. Для этого проверим, как текст бьется на токены. Распечатаем первые пять.

```
doc.segment(segmenter)
display(doc.tokens[:5])
```

```
[]: [DocToken(stop=8, text='ПРИГОВОР'),

 DocToken(start=9, stop=15, text='ИМЕНЕМ'),
 DocToken(start=17, stop=27, text='РОССИЙСКОЙ'),
 DocToken(start=28, stop=37, text='ФЕДЕРАЦИИ'),
 DocToken(start=38, stop=40, text='01')]
```

Посмотрим, как библиотека разбивает текст на предложения. Проверим первые пять.

```
display(doc.sents[:5])
#(print – выводит в строчку; display – в столбец)
```

```
[]: [DocSent(stop=1999, text='ПРИГОВОР\nИМЕНЕМ  РОССИЙСКОЙ ФЕДЕРАЦИИ\n01
декабр..., tokens=[...]),

 DocSent(start=2001, stop=3080, text='Во исполнение своего совместного
преступного план..., tokens=[...]),
 DocSent(start=3081, stop=6135, text='Продолжая реализовывать совместный с
соучастникам..., tokens=[...]),
 DocSent(start=6137, stop=7085, text='Таким образом, ************ А.А. и его
неустановл..., tokens=[...]),
```

```
 DocSent(start=7087, stop=7939, text='Суд, проведя судебное следствие,
выслушав судебны..., tokens=[...])]
```

Библиотека умеет делать морфологически разбор слов. Это может пригодиться, если для какой-то задачи вам понадобиться вычленять и работать только, например, с прилагательными или существительными. Давайте опять посмотрим на разбор первых пяти слов.

```
doc.tag_morph(morph_tagger)
display(doc.tokens[:5])
```

```
[]: [DocToken(stop=8, text='ПРИГОВОР', pos='NOUN',
feats=<Inan,Nom,Masc,Sing>),

 DocToken(start=9, stop=15, text='ИМЕНЕМ', pos='PROPN',
feats=<Anim,Gen,Masc,Sing>),
 DocToken(start=17, stop=27, text='РОССИЙСКОЙ', pos='ADJ',
feats=<Gen,Pos,Fem,Sing>),
 DocToken(start=28, stop=37, text='ФЕДЕРАЦИИ', pos='PROPN',
feats=<Inan,Gen,Fem,Sing>),
 DocToken(start=38, stop=40, text='01', pos='ADJ')]
```

Как видим, библиотека дает полный разбор. У каждого слова указана часть речи, а также, например, в каком числе, роде и т.д. это слово употребляется.

Библиотека Natasha умеет и нормализовывать слова – приводить их к правильной форме. И не только слова, но и целые словосочетания, которые мы обычно воспринимаем как цельную сущность. Давайте посмотрим на небольшую выборку.

```
for span in doc.spans:
    span.normalize(morph_vocab)
```

```
{_.text: _.normal for _ in doc.spans if _.text != _.normal}
```

```
[]: {'А.А.\nДопрошенный': 'А.А. Допрошенный',

 'Андрея Александровича': 'Андрей Александрович',
 'Биткова А.Ю.': 'Битков А.Ю.',
 'Давыдовой Д.Е.': 'Давыдова Д.Е.',
 'Кормилицыну Д.А.': 'Кормилицын Д.А.',
 'Москве': 'Москва',
 'Москвы': 'Москва',
 'Попова А.В.': 'Попов А.В.',
 'Правительства': 'Правительство',
 'России': 'Россия',
 'Российской Федерации': 'Российская Федерация',
 'Русинов А.А.': 'Русины А.А.',
 'Русинова А.А.': 'Русинов А.А.',
 'Русиновым А.А.': 'Русинов А.А.',
 'Службы': 'Служба',
 'Службы по': 'Служба по'}
```

Как видите, библиотека воспринимает, например, «Российской Федерации» не как два отдельных слова «Российская» и «Федерация», а как цельную сущность. А также умеет приводить это к правильной форме – «Российская Федерация». Также происходит и с именами. Библиотека может распознавать их род и приводить к начальной форме.

Если вам нужно привести каждое слово к начальной форме, то есть лемматизировать его, то вы можете использовать команду `lemmatize`.

```
for token in doc.tokens:
    token.lemmatize(morph_vocab)
```

```
{_.text: _.lemma for _ in doc.tokens}
```

```
[]: …….
```

```
 '»': '»',

 'А': 'а',
 'Александровича': 'александрович',
 'Андрея': 'андрей',
 'Билайн': 'билайн',
 'Биткова': 'битков',
 'Буженинова': 'буженинов',
 'В': 'в',
 'ВАО': 'вао',
 'Вещественное': 'вещественный',
 'Во': 'в',
 'Горшковой': 'горшков',
 'Д': 'д',
 'Давыдовой': 'давыдов',
 'Далее': 'далее',
 'Допрошенный': 'допросить',
 'Дудукова': 'дудукова',
```

Библиотека Natasha умеет извлекать даты. Достанем даты из приговора:

```
matches = dates_extractor(text)
facts = [i.fact.as_json for i in matches]
facts
```

```
[]: [OrderedDict([('year', 2016), ('month', 12), ('day', 1)]),

 OrderedDict([('year', 1998), ('month', 1), ('day', 8)]),
 OrderedDict([('year', 2015), ('month', 9), ('day', 16)]),
 OrderedDict([('year', 2015), ('month', 9), ('day', 16)]),
 OrderedDict([('year', 1998), ('month', 1), ('day', 8)]),
 OrderedDict([('year', 2015), ('month', 9), ('day', 16)]),
 OrderedDict([('year', 2015), ('month', 9), ('day', 16)]),
 OrderedDict([('year', 2015), ('month', 9), ('day', 16)]),
 OrderedDict([('year', 2015), ('month', 10), ('day', 16)]),
 OrderedDict([('year', 1998), ('month', 6), ('day', 30)]),
 OrderedDict([('year', 2016), ('month', 4), ('day', 1)]),
```

```
 OrderedDict([('year', 2012), ('month', 10), ('day', 1)]),
 OrderedDict([('year', 2015), ('month', 9), ('day', 16)]),
```

Приведем это к более удобной форме с помощью f-строк:

```
for f in facts:
    print(f"{f.get('day')}.{f.get('month')}.{f.get('year')}")
```

```
[]: 1.12.2016

8.1.1998
16.9.2015
16.9.2015
8.1.1998
16.9.2015
16.9.2015
16.9.2015
16.10.2015
30.6.1998
```

Библиотека может извлекать имена в разных написаниях из текстов. В приговорах чаще – это фамилия и инициалы.

```
for span in doc.spans:
    if span.type == nt.PER:
        span.extract_fact(names_extractor)
names_dict = {_.normal: _.fact.as_dict for _ in doc.spans
if _.fact}
names_dict
```

```
[]: {'А.А': {'last': 'А'},

 'А.А.': {'last': 'А'},
 'А.А. Допрошенный': {'first': 'А', 'last': 'Допрошенный', 'middle': 'А'},
 'Андрей Александрович': {'first': 'Андрей', 'last': 'Александрович'},
 'Битков А.Ю.': {'first': 'А', 'last': 'Битков', 'middle': 'Ю'},
 'Горшковой А.А.': {'first': 'А', 'last': 'Горшковой', 'middle': 'А'},
 'Давыдова Д.Е.': {'first': 'Д', 'last': 'Давыдова', 'middle': 'Е'},
 'Дудукова М.Ю.': {'first': 'М', 'last': 'Дудукова', 'middle': 'Ю'},
 'Духновская З.А.': {'first': 'З', 'last': 'Духновская', 'middle': 'А'},
```

Как видим, каждая часть имени распознана отдельно: есть фамилия, имя и отчество. Нам нужно взять распознанную сущность целиком. Поэтому с помощью команды `keys()` мы можем получить все ключи словаря, а дальше сделать из него список.

```
list(names_dict.keys())
```

```
[]: ['Духновской З.А.',
```

```
'Горшковой А.А.',
'Давыдова Д.Е.',
'Дудукова М.Ю.',
'Чубенко А.А.',
'А.А.',
'Попов А.В.',
'Колокольниковой Е.В.',
```

При исследовании текста попутно распознаются и лишние элементы. Но это все легко вычистить на этапе очистки данных. Главное, чтобы ничего не потерять при этом.

При проведении расследования этап извлечения имен из текстов приговоров на этом был закончен. Дальше оставалось только проверить встречаются ли эти имена и в других делах.

Результатами практического занятия являются:

1) знакомство с библиотекой natasha;
2) разбор базовых задач обработки естественного русского языка: сегментации на токены и предложения, морфологический и синтаксический анализ, лемматизация, извлечение именованных сущностей.

Уважаемые слушатели! Проделайте все шаги самостоятельно и прикрепите файл или ссылку на Colab.