Практика 2.3.2. Предварительная обработка текста на Python: инструменты и алгоритмы (Часть 1)

На втором практическом занятии будем подробно обсуждать основные этапы предварительной обработки текста с целью преобразования текстовой информации с человеческого языка в машиночитаемый формат для последующей обработки. Кроме того, дополнительно обсудим инструменты, необходимые для процесса предварительной обработки текста.

Общие шаги регуляризации (обработки) текста включают в себя:

- преобразование всех букв в тексте в строчные или прописные буквы;
- преобразование чисел в тексте в слова или удаление этих чисел;
- удаление знаков препинания, ударений и других знаков, которые находятся в тексте;
- удаление пробелов в тексте;
- сокращения в расширенном тексте;
- удаление стоп-слов, разреженных слов и других определенных слов, которые находятся в тексте;
- нормализация текста (text canonicalization).

Рассмотрим подробно шаги алгоритма регуляризации текста. Следующие маленькие скрипты следует рассматривать как «строительные блоки» для анализа текста и предварительной обработки данных.

Пример 1. Работа с регистрами

Преобразование букв в нижний регистр

Для преобразования символов строки к нижнему или верхнему регистру используют функции `lower()` или `upper()`.

Код реализации Python:

```
input_str = "В пятерку крупнейших стран по численности населения в 2020 году входили Китай, Индия, США, Индонезия и Бразилия."
# Приведем к нижнему регистру
input_str = input_str.lower()
print(input_str)
```

[1]: `в пятерку крупнейших стран по численности населения в 2020 году входили китай, индия, сша, индонезия и бразилия.`

Преобразование букв в верхний регистр

```
input_str = "В пятерку крупнейших стран по численности населения в 2020 году входили Китай, Индия, США, Индонезия и Бразилия."
# Приведем к верхнему регистру
input_str = input_str.upper()
print(input_str)
```

[2]: `В ПЯТЕРКУ КРУПНЕЙШИХ СТРАН ПО ЧИСЛЕННОСТИ НАСЕЛЕНИЯ В 2020 ГОДУ ВХОДИЛИ КИТАЙ, ИНДИЯ, США, ИНДОНЕЗИЯ И БРАЗИЛИЯ.`

Пример 2. Удалить пробелы в тексте

Удалить пробелы в начале и конце текста можно с помощью функции `strip()`.

Код реализации Python:

```
input_str = " \t   удалим пробелы в начале и конце текста \t "
input_str = input_str.strip()
input_str
```

[3]: `'удалим пробелы в начале и конце текста'`

Пример 3. Удалить пунктуацию в тексте

В данном примере кода покажем, как удалить знаки препинания, ударения в тексте, например, [! »# $% &’ () * +, -. / :; <=>? @ [\] ^ _ `{|} ~] и другие символы. Для этого воспользуемся модулем `string`, который предоставляет атрибут со всеми символами пунктуации – `string.punctuation`.

Константа с именем `string.punctuation` обеспечивает большой список знаков препинания:

```
print(string.punctuation)
```

[4]: `!"#$%&'()*+,-./:;<=>?@[\]^_`{|}~`

Рассмотрим примеры для русского и английского предложений:

```
# Пример 3.1
import string
input_str = "- Мама мыла &раму, [а] Маша. помогала?. (из прописи 1 класса). !!!"
result = input_str.translate(str.maketrans("", "", string.punctuation))
print(result)
```

[3_1]: `Мама мыла раму а Маша помогала из прописи 1 класса`

```
# Пример 3.2
import string
input_str = "This &is [an] example? {of} string. with.? pun
ctuation!!!!"
result = input_str.translate(str.maketrans("", "", string.p
unctuation))
print(result)
```

[3_2]: `This is an example of string with punctuation`

Пример 4. Очистка строки

Теперь мы можем применить информацию из предыдущих пунктов для очистки строки. Это один из наиболее востребованных процессов в проектах data science при очистке данных. Данный пример – это необработанный текст с пробельными символами и переносами строк. Ниже приведен простой скрипт для очистки такой строки:

Код реализации Python:

```
# Текстовая строка с «мусором»
test_string_with_garbage = 'The quick brown
fox\njumps\tover the\tlazy dog\r\n'
character_map = {
ord('\n') : ' ',
ord('\t') : ' ',
ord('\r') : None
}
test_string_with_garbage.translate(character_map)
```

```
Out[4]: 'The quick brown fox jumps over the lazy dog'
```

Пример 5. Разбиение строки

Для анализа текста требуются различные метрики, такие как количество слов, количество символов, средняя длина предложения. Чтобы вычислить эти значения, нам нужно подготовить текст – очистить и разделить. В Python есть несколько встроенных функций для разделения текста.

Функция `split()` разбивает строку на подстроки по разделителю:

`str.split([разделитель [, maxsplit]]).`

Параметр `maxsplit` позволяет указать максимально количество разделений. Если разделитель не задать, то по умолчанию будет выбрано значение пробела.

- Разбиение по пробелу (по умолчанию):

```
test_string.split()
```

```
Out[5]: ['The', 'quick', 'brown', 'fox', 'jumps', 'over',
'the', 'lazy', 'dog']
```

- Разбиение по определенному числу пробелов:

```
# текст будет разделен по 2-м пробелам; количество
элементов в списке = maxsplit+1
test_string.split(' ', 2)
```

```
Out[6]: ['The', 'quick', 'brown fox jumps over the lazy
dog']
```

- Разбиение по произвольному символу:

```
test_string.split('e')
```

```
Out[7]: ['Th', ' quick brown fox jumps ov', 'r the lazy
dog']
```

- Разбиение строки по нужному токену с токенами до и после него:

```
test_string.partition('fox')
```

```
Out[8]: ('The quick brown ', 'fox', ' jumps over the lazy
dog')
```

Пример 6. Токенизация строки

Токен – это последовательность символов в документе. **Токенизация** – разбиение текста на слова (и не только на слова: знаки препинания, границы абзацев и т.п).

Соберем все, что мы узнали ранее, и применим это для токенизации с использованием pandas. В примере мы должны очистить строку от лишних символов, привести к одному регистру и разбить ее на токены.

```
import pandas as pd
test_punctuation = " This &is [an] example? {of} string.
with.? punctuation!!!! "
data = pd.DataFrame([test_punctuation])
data.iloc[0].str.lower().str.replace('\W+', '
').str.strip().str.split()
```

```
Out[9]: [this, is, an, example, of, string, with,
punctuation]
Name: 0, dtype: object
```

Пример 7. Подсчитать количество слов в предложении и количество букв в слове

Количество слов в предложении (знаки препинания не учитываются!):

```
input_str = "В пятерку крупнейших стран по численности насе
ления в 2020 году входили Китай, Индия, США, Индонезия и Бр
азилия."
words = input_str.split()
print('Number of words in text file :', len(words))

Out[10]: Number of words in text file : 17
```

Количество букв в слове:

```
import collections
dict(collections.Counter('aaaarggh'))

Out[11]: {'a': 4, 'g': 2, 'h': 1, 'r': 1}
```

или

```
from collections import Counter
print(Counter("aaaarggh"))

Out[12]: Counter({'a': 4, 'g': 2, 'r': 1, 'h': 1})
```

Пример 8. Удалить стоп-слова в тексте

Стоп слова (stop words) – наиболее распространенные слова в языках, например, в английском: «the», «a», «open», «is», «all». Эти слова не имеют специального или важного значения и обычно могут быть удалены из текста. Для этого обычно используется библиотека Natural Language ToolKit (NLTK), предназначенная для статистики символов и обработки естественного языка.

Код реализации Python:

```
# Пример 8.1
import nltk
nltk.download('stopwords')
from nltk.corpus import stopwords
# импорт стоп-слов
nltk.download('punkt')

input_str = "NLTK is a leading platform for building Python
programs to work with human language data."
```

```
stop_words = set(stopwords.words('english'))
from nltk.tokenize import word_tokenize
tokens = word_tokenize(input_str)
result = [i for i in tokens if not i in stop_words]
print(result)
```

```
Out[13]: ['NLTK', 'leading', 'platform', 'building', 'Python', 'programs',
'work', 'human', 'language', 'data', '.']
```

```
# Пример 8.2
import nltk
nltk.download('stopwords')
from nltk.corpus import stopwords
# импорт стоп-слов
nltk.download('punkt')

input_str = "Аз есмь строка, живу я, мерой остр. За семь мо
рей ростка я вижу рост. Я в мире — сирота. Я в Риме —
 Ариост."
stop_words = set(stopwords.words('russian'))
from nltk.tokenize import word_tokenize
tokens = word_tokenize(input_str)
result = [i for i in tokens if not i in stop_words]
print(result)
```

```
Out[14]: ['Аз', 'есмь', 'строка', ',', 'живу', ',', 'мерой', 'остр', '.',
'За', 'семь', 'морей', 'ростка', 'вижу', 'рост', '.', 'Я', 'мире', '—',
'сирота', '.', 'Я', 'Риме', '—', 'Ариост', '.']
```

Пример 9. Стемминг (Stemming)

Stem – корень; **Stemming** – извлечение основы – это процесс, который сводит слова к основам, корням или формам, например, books–book, looked–look). Существует два основных алгоритма: Porterstemming алгоритм и Lancasterstemming алгоритм.

Использование **NLTK** для stemming

```
from nltk.stem import PorterStemmer
from nltk.tokenize import word_tokenize
stemmer = PorterStemmer()
input_str = "There are several types of stemming algorithms."
input_str = word_tokenize(input_str)
for word in input_str:
    print(stemmer.stem(word))
Out[15]:
```

```
there
are
sever
type
of
stem
algorithm
```

Лемматизации

Ле́мма (англ. lemma) – начальная, словарная форма слова. В русском языке для существительных и прилагательных это форма именительного падежа единственного числа, для глаголов и глагольных форм – форма инфинитива. Процесс автоматического приведения слов текста к леммам называется лемматизация.

Например, фраза "*Утром рано мама Милы мыла раму мылом*" после лемматизации будет выглядеть так: *утро, рано, мама, Мила, мыть, рама, мыло*.

Лемматизация – это процесс преобразования слова в его базовую форму. Разница между стемминг (stemming) и лемматизацией заключается в том, что лемматизация учитывает контекст и преобразует слово в его базовую форму, тогда как стемминг просто удаляет последние несколько символов, что часто приводит к неверному значению и орфографическим ошибкам. Лемматизация использует словарную базу знаний для получения правильной формы слова.

Например, лемматизация правильно определила бы базовую форму «caring» и «care», в то время как стемминг отрезал бы «ing» и преобразовал ее в car.

```
«Caring» -> Лемматизация -> «Care»
«Caring» -> Стемминг -> «Car»
```

Реализация лемматизации может осуществляться с помощью следующих пакетов Python.

- Wordnet Lemmatizer
- Spacy Lemmatizer
- TextBlob
- CLiPS Pattern
- Stanford CoreNLP
- Gensim Lemmatizer
- TreeTagger

Пример 10: Использование NLTK для лемматизации

Лемматизатор Wordnet из NLTK

Wordnet – это большая, свободно распространяемая и общедоступная лексическая база данных для английского языка с целью установления

структурированных семантических отношений между словами. Библиотека также предлагает возможности лемматизации и является одним из самых ранних и наиболее часто используемых лемматизаторов.

Пример 10.1

```
# В начале необходимо загрузить библиотеку
import nltk
nltk.download('wordnet')
nltk.download('punkt')

# Для того, чтобы лемматизировать, нужно создать экземпляр
WordNetLemmatizer() и вызвать функцию lemmatize():

from nltk.stem import WordNetLemmatizer
from nltk.tokenize import word_tokenize
# Init the Wordnet Lemmatizer
lemmatizer = WordNetLemmatizer()
input_str = "been had done languages cities mice"
input_str = word_tokenize(input_str)
for word in input_str:
    print(lemmatizer.lemmatize(word))
```

```
been
had
done
language
city
mouse
```

Давайте лемматизируем другое предложение. Сначала разобъем предложение на слова с помощью `nltk.word_tokenize`, а затем вызывём `lemmatizer.lemmatize()` для каждого слова.

Пример 10.2

```
# Define the sentence to be lemmatized
sentence = "The striped bats are hanging on their feet for
best"
# Tokenize: Split the sentence into words
word_list = nltk.word_tokenize(sentence)
print(word_list)
```

```
['The', 'striped', 'bats', 'are', 'hanging', 'on', 'their', 'feet',
'for', 'best']
```

Пример 10.3

```
# Lemmatize list of words and join
```

```
lemmatized_output = ' '.join([lemmatizer.lemmatize(w) for w
in word_list])
print(lemmatized_output)
```

```
[Out]: The striped bat are hanging on their foot for best
```

Обратите внимание, что есть ошибки. Потому что «are» не преобразовалось в «be», а «Hang» не преобразовалось в «hang», как ожидалось. Это можно исправить, если в качестве второго аргумента `lemmatize()` указать правильный тег «part-of-speech» (POS-тег).

Пример 11: Иногда одно и то же слово может иметь несколько лемм в зависимости от значения / контекста:

```
print(lemmatizer.lemmatize("stripes", 'v'))
#[Out]: strip

print(lemmatizer.lemmatize("stripes", 'n'))
#[Out]: stripe
```

Пример 12: Wordnet Lemmatizer с соответствующим POS-тегом

Достаточно сложно вручную проставить соответствующий POS-тег для каждого слова при обработке больших текстов. Поэтому вместо этого мы найдем правильный POS-тег для каждого слова, сопоставим его с правильным входным символом, который принимает WordnetLemmatizer, и передадим его в качестве второго аргумента в `lemmatize()`.

Как получить POS-тег для выбранного слова?

В nltk для этого есть метод `nltk.pos_tag()`. Он принимает только список (список слов), даже если нужно передать только одно слово.

```
# Пример 12.1
import nltk
nltk.download('averaged_perceptron_tagger')
print(nltk.pos_tag(['feet']))
# [Out]: [('feet', 'NNS')]

# Пример 12.2
sentence = "The striped bats are hanging on their feet for best"
print(nltk.pos_tag(nltk.word_tokenize(sentence)))
# [Out]: [('The', 'DT'), ('striped', 'JJ'), ('bats', 'NNS'), ('are',
'VBP'), ('hanging', 'VBG'), ('on', 'IN'), ('their', 'PRP$'),
('feet', 'NNS'), ('for', 'IN'), ('best', 'JJS')]
```

`nltk.pos_tag()` возвращает кортеж с тегом POS. Ключевым моментом здесь является сопоставление POS-тегов NLTK с форматом, принятым лемматизатором wordnet.

```
# Пример 12.3
# Lemmatize with POS Tag
from nltk.corpus import wordnet
def get_wordnet_pos(word):
    """Map POS tag to first character lemmatize() accepts"""
    tag = nltk.pos_tag([word])[0][1][0].upper()
    tag_dict = {"J": wordnet.ADJ,
                "N": wordnet.NOUN,
                "V": wordnet.VERB,
                "R": wordnet.ADV}
    return tag_dict.get(tag, wordnet.NOUN)

# 1. Init Lemmatizer
lemmatizer = WordNetLemmatizer()

# 2. Lemmatize Single Word with the appropriate POS tag
word = 'feet'
print(lemmatizer.lemmatize(word, get_wordnet_pos(word)))

# 3. Lemmatize a Sentence with the appropriate POS tag
sentence = "The striped bats are hanging on their feet for best"
print([lemmatizer.lemmatize(w, get_wordnet_pos(w)) for w in
nltk.word_tokenize(sentence)])

# [Out]: ['The', 'strip', 'bat', 'be', 'hang', 'on', 'their',
'foot', 'for', 'best']
```

Пример 13: spaCy лематтизация

spaCy является относительно новым пакетом и на данный момент считается стандартом в индустрии NLP. Он поставляется с предварительно созданными моделями, которые могут анализировать текст и выполнять различный функционал, связанный с NLP. spaCy по умолчанию определяет часть речи и назначает соответствующую лемму.

```
import spacy
# Initialize spacy 'en' model, keeping only tagger component needed
for lemmatization
nlp = spacy.load('en', disable=['parser', 'ner'])
sentence = "The striped bats are hanging on their feet for best"
# Parse the sentence using the loaded 'en' model object `nlp`
doc = nlp(sentence)
# Extract the lemma for each token and join
" ".join([token.lemma_ for token in doc])

#[Out]: 'the strip bat be hang on -PRON- foot for good'
```

spaCy выполнил все лемматизации, которые так же выполнил лемматизатор Wordnet, с проставленным правильным тегом POS. Плюс к этому также лемматизируется такое слово как «best» превращаясь в «good».

Пример 13: TextBlob Lemmatizer

TextBlob – это мощный, быстрый пакет NLP. Используя объекты Word и TextBlob, довольно просто анализировать и лемматизировать слова и предложения соответственно.

```
# Пример 13.1
# pip install textblob
from textblob import TextBlob, Word
# Lemmatize a word
word = 'stripes'
w = Word(word)
w.lemmatize()

# [Out]: stripe
```

Чтобы лемматизировать предложение или абзац, мы анализируем его с помощью TextBlob а затем вызвать функцию `lemmatize()` для проанализированных слов.

Lemmatize a sentence

```
# Пример 13.2
sentence = "The striped bats are hanging on their feet for
best"
sent = TextBlob(sentence)
" ". join([w.lemmatize() for w in sent.words])

# [Out]: 'The striped bat are hanging on their foot for
best'
```

Видно, что TextBlob не справился с работой. Но, как и NLTK, TextBlob внутри использует Wordnet. Соответственно, для корректной работы так же требует передачу соответствующего POS-тега методу `lemmatize()`.

```
# Пример 13.3 TextBlob Lemmatizer с соответствующим POS-
тегом
# Define function to lemmatize each word with its POS tag
def lemmatize_with_postag(sentence):
    sent = TextBlob(sentence)
    tag_dict = {"J": 'a',
                "N": 'n',
                "V": 'v',
                "R": 'r'}
```

```
    words_and_tags = [(w, tag_dict.get(pos[0], 'n')) for w,
pos in sent.tags]
    lemmatized_list = [wd.lemmatize(tag) for wd, tag in
words_and_tags]
    return " ".join(lemmatized_list)
# Lemmatize
sentence = "The striped bats are hanging on their feet for
best"
lemmatize_with_postag(sentence)

# [Out]:  'The striped bat be hang on their foot for best'
```

Пример 14: Сравнение NLTK, TextBlob, spaCy

Давайте запустим лемматизацию, используя 3 пакета для следующих предложений, и сравним вывод.

```
# Пример 14.1
sentence = """Following mice attacks, caring farmers were
marching to Delhi for better living conditions.
Delhi police on Tuesday fired water cannons and teargas
shells at protesting farmers as they tried to
break barricades with their cars, automobiles and
tractors."""

# NLTK
from nltk.stem import WordNetLemmatizer
lemmatizer = WordNetLemmatizer()
import string

print(" ".join([lemmatizer.lemmatize(w, get_wordnet_pos(w))
for w in nltk.word_tokenize(sentence) if w not in
string.punctuation]))

# ('Following mouse attack care farmer be march to Delhi
for well living '
# 'condition Delhi police on Tuesday fire water cannon and
teargas shell at '
# 'protest farmer a they try to break barricade with their
car automobile and '
# 'tractor')

# Пример 14.2
# Spacy
import spacy
nlp = spacy.load('en', disable=['parser', 'ner'])
```

```
doc = nlp(sentence)
print(" ".join([token.lemma_ for token in doc]))

# ('follow mice attack , care farmer be march to delhi for
good living condition '
# '. delhi police on tuesday fire water cannon and teargas
shell at protest '
# 'farmer as -PRON- try to break barricade with -PRON- car
, automobile and '
# 'tractor .')

# Пример 14.3
# TextBlob
print(lemmatize_with_postag(sentence))

# ('Following mouse attack care farmer be march to Delhi
for good living '
# 'condition Delhi police on Tuesday fire water cannon and
teargas shell at '
# 'protest farmer a they try to break barricade with their
car automobile and '
# 'tractor')
==========================================================
```

Для дальнейшей работы нам понадобится понятие – **регулярные выражения**, которые мы рассмотрим в части 2 практического занятия **2.3.2. Предварительная обработка текста на Python: инструменты и алгоритмы**.