Авторы:

Трапезников Алексей Александрович

студент

Ардаева Анастасия Андреевна

студентка

ФГБОУ ВО «Восточно-Сибирский государственный

университет технологий и управления»

г. Улан-Удэ, Республика Бурятия

АЛГОРИТМ ПРЕПРОЦЕССОРНОЙ ОБРАБОТКИ ЕЯ-ТЕКСТА НА PYTHON

Аннотация*: в статье рассматривается препроцессорная обработка естественно-языкового текста, которая является важным этапом подготовки текста к решению различных задач: машинный перевод, автореферирование, понимание смысла текста и многих других. В работе рассмотрены основные этапы подготовки текста: токенизация, фильтрация и нормализация. Полученный результат может использоваться для построения векторной модели текста и в дальнейшем в разных алгоритмах машинного обучения.*

Ключевые слова*: естественный язык, обработка ЕЯ-текста, токенизация, фильтрация, нормализация, Python, NLTK, Pymorphy2.*

Введение. В настоящее время обработка естественного языка не используются разве что в совсем консервативных отраслях. В большинстве же технологических решений распознавание и обработка «человеческих» языков обязательно. Любая задача по обработке естественно-языкового текста (Natural Language Processing – NLP) требует его предварительной подготовки, конечная цель которой получение нормализованного текста без шума.

В работе алгоритмы написаны на высокоуровневом языке программирования Python, имеющем динамическую семантику. Он является широко используемым языком в машинном обучении. Причина такой популярности в наборе предварительно настроенных инструментов.

Благодаря простому синтаксису в сравнении с другими языками прост и легко читаем, соответственно время написания программ сокращается. Для данного языка написано множество библиотек в том числе и для обработки текстов. Самыми популярными являются NLTK, Pymorphy2 и Codecs.

NLTK (Natural Language Toolkit) – это пакет библиотек и программ для символьной и статистической обработки естественного языка. NLTK предоставляет открытый доступ для ряда инструментов и подходит для изучения: компьютерной лингвистики, искусственного интеллекта, информационного поиска и машинного обучения [3].

Библиотека Pymorphy2 – морфологический анализатор для русского языка, использующий словари из открытого корпуса русского языка Open-Corpora [1].

Библиотека Codecs определяет базовые классы стандартных кодеков Python. Данная библиотека необходима для открытия файла с помощью функции codecs.open(), который открывает закодированный файл в заданном режиме и в заданной кодировке [2].

Приведенные выше библиотеки используются для разных этапов предпроцессорной обработки.

Этапы предпроцессорной обработки

Технология предпроцессорной обработки ЕЯ-текста состоит из трех этапов. На рисунке 1 представлен алгоритм предпроцессорной обработки ЕЯ-текста. Кратко рассмотрим каждый этап.

Первый этап: лексический анализ, или токенизация. Токенизация означает разделение текста на токены. А токен – это объект, создающийся из лексемы в процессе лексического анализа, то есть токенизации. В роли токена могут выступать слова и знаки препинания.

Второй этап: фильтрация, другими словами, удаление «стоп-слов» и знаков препинаний. «Стоп-слова» – это те фразы и слова, которые не несут смысловую нагрузку [5]. В русском языке к таким словам можно отнести предлоги, союзы и частицы. Их удаляют для достижения лучшего результата работы с текстом.

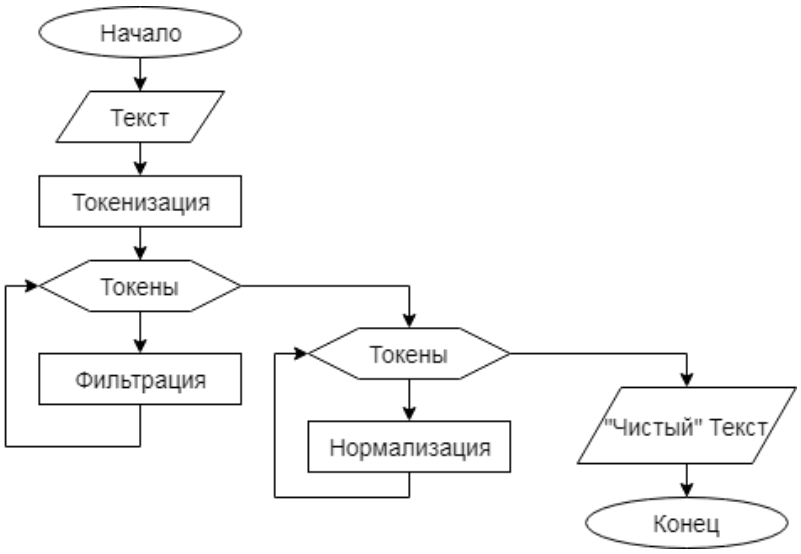

Рис. 1. Алгоритм предпроцессорной обработки

Третий этап: нормализация. Одно и то же слово может быть записано в разной форме и для определения его частоты встречаемости необходимо знать его нормальную форму.

Реализация алгоритма

Рассмотрим более детально алгоритм препроцессорной обработки на примере.

Первый этап алгоритма: токенизация.

Исходные данные: переменная text – текст, состоящий минимум из одного предложения.

Для токенизации была использована функция word_tokenize() из библиотеки NLTK языка Python. Модуль tokenize представляет собой лексический сканер для исходного кода на Python. Сканер этого модуля возвращает комментарии как токены [4].

Фрагмент данной функции представлен в листинге 1.

Листинг 1 – Функция word_tokenize()

```
from nltk.tokenize import word_tokenize

text = «Токенизация означает разделение текста на токены. А токен – это объект, создающийся из лексемы в процессе лексического анализа, то есть токенизации. В роли токена могут выступать слова и знаки препинания.»

tokens = word_tokenize(text)
```

print(tokens)

Результат работы функции показан на рис. 2.

```
['Токенизация', 'означает', 'разделение', 'текста', 'на', 'токены', '.',
'А', 'токен', '-', 'это', 'объект', ',', 'создающийся', 'из', 'лексемы', 'в',
'процессе', 'лексического', 'анализа', ',', 'то', 'есть', 'токенизации', '.',
'В', 'роли', 'токена', 'могут', 'выступать', 'слова', 'и', 'знаки', 'препинания', '.']
```

Рис. 2. Результат функции word_tokenize()

Как видно из рисунка 2 данная функция разделила текст на токены и записала в list, с помощью которого можно работать с каждым токеном отдельно. Это облегчит фильтрацию и дальнейшую работу с текстом.

Второй этап алгоритма: фильтрация.

Исходные данные: переменная tokens – текст, разделенный на токены.

Для фильтрации были определены «стоп-слова» русского языка из библиотеки NLTK с помощью функции stopwords.words('russian') из класса stopwords и дополнены знаками препинания.

Краткий список «стоп-слов» из библиотеки NLTK = {и в во не что он на я с со как а то все она так его … }

Знаки препинания = {., – - _ = + /! ” ; : %? * ( )№ < > " `` – ^ & @ « » '\'\" $ # { } [ ] '}

Фрагмент листинга данного этапа представлен в листинг 2

Листинг 2 – Функция фильтрации текста

```
def get_stop_words(self): #получить список стоп-слов
with codecs.open(«stop_words», «r», «utf_8_sig») as f:
stop_words = f.read()
stop_words = word_tokenize(stop_words)
stop_words += stopwords.words('russian')
return stop_words
def word_processing(self): #препроцессорная обработка текста
# токенизация
tokens = word_tokenize(self.text)
```

```
# удаление стоп-слов
stop_words = self.get_stop_words()
tokens = [word for word in tokens if (word not in stop_words)]
return tokens
```

Как видно из листинга 2 предпроцессорная обработка текста происходит в методе word_processing(). А метод get_stop_words() разработан для получения «стоп-слов».

На вход процесса фильтрации был представлен результат выполнения функции word_tokenize() (рисунок 2).

```
['Токенизация', 'означает', 'разделение', 'текста', 'токены', 'А',
'токен', 'это', 'объект', 'создающийся', 'лексемы', 'процессе',
'лексического', 'анализа', 'токенизации', 'В', 'роли', 'токена',
'могут', 'выступать', 'слова', 'знаки', 'препинания']
```

Рис. 3. Результат фильтрации текста

И как видно на рисунке 3 в результате в тексте были удалены все «стоп-слова» и знаки препинания. Это можно проверить, сравнив рисунок 2 и рисунок 3. Таким образом, после выполнения данного этапа текст «чист» и можно дальше работать с текстом без лишних слов и символов.

Заключительный этап – нормализация (постановка слов в начальную форму.

Для данной цели необходимо использовать библиотеку Pymorphy2. Нормальную (начальную) форму слова можно получить через атрибут Parse.normal_form и Parse.normalized. Например: morph.parse('думающему')[0].normal_form результатом будет 'думать'.

Таким образом, все три этапа алгоритма предпроцессорной обработки можно быстро решить благодаря возможностям языка Python и используемых библиотек.

Заключение. В работе рассмотрен подготовительный процесс обработки текста на естественном языке. Надо отметить, что высокоуровневый язык программирования Python имеет много библиотек, которые позволяют значительно упростить работу с текстом.

Список литературы

1. Mikhail Korobov. Морфологический анализатор pymorphy2 [Электронный ресурс]. – Режим доступа: https://pymorphy2.readthedocs.io/en/latest/index.html (дата обращения: 18.05.2019).

2. Python [Электронный ресурс]. – Режим доступа: https://www.python.org/about/ (дата обращения: 14.05.2019).

3. Bird S., Klein E., and Loper E. Natural Language Processing with. – 2009. – С. 221–261.

4. Tokenize [Электронный ресурс]. – Режим доступа: http://python-lab.ru/documentation/27/stdlib/tokenize.html (дата обращения: 14.05.2019).

5. Что такое стоп-слова [Электронный ресурс]. – Режим доступа: https://semantica.in/blog/chto-takoe-stop-slova.html (дата обращения: 14.04.2019).