Научная статья
УДК 37
DOI: 10.25688/2072-9014.2024.68.2.05

ИЗУЧЕНИЕ МЕТОДОВ ОБРАБОТКИ ЕСТЕСТВЕННОГО ЯЗЫКА С ИСПОЛЬЗОВАНИЕМ ЯЗЫКА ПРОГРАММИРОВАНИЯ PYTHON И БИБЛИОТЕКИ SPACY В ВЫСШЕЙ ШКОЛЕ

Иван Юрьевич Пикалов

Курский государственный университет,
Курск, Россия
pikalov@kursksu.ru

Аннотация. В статье приводится обоснование необходимости изучения методов обработки естественного языка (*англ.* Natural Language Processing, NLP) в высшей школе, предлагается содержание соответствующего раздела учебного курса и методика его изучения. Работа может быть использована при обучении студентов разных направлений подготовки, будет способствовать пониманию принципов обработки текстов на естественном языке. Предлагаемые инструменты и методы можно использовать при решении задач обработки естественного языка в профессиональной деятельности.

Ключевые слова: обработка естественного языка; обработка текста; искусственный интеллект; NLP; SpaCy.

Original article
UDC 37
DOI: 10.25688/2072-9014.2024.68.2.05

LEARNING NATURAL LANGUAGE PROCESSING TECHNIQUES USING THE PYTHON PROGRAMMING LANGUAGE AND THE SPACY LIBRARY IN HIGH SCHOOL

Ivan Yu. Pikalov

Kursk State University,
Kursk, Russia
pikalov@kursksu.ru

Abstract. The paper explains the necessity of teaching natural language processing (Natural Language Processing, NLP) in higher education institutions and gives the relevant course overview. This course can be incorporated into different training programm tracks and allow students understand the basics of natural language processing. The proposed methods and tools can be used to solve tasks of natural language processing in professional sphere.

Keywords: natural language processing; text processing; artificial intelligence; NLP; SpaCy.

© Пикалов И. Ю., 2024

Для цитирования: Пикалов И. Ю. Изучение методов обработки естественного языка с использованием языка программирования Python и библиотеки SpaCy в высшей школе / И. Ю. Пикалов // Вестник МГПУ. Серия «Информатика и информатизация образования». 2024. № 2 (68). С. 48–61.

For citation: Pikalov I. Yu. Learning natural language processing techniques using the Python programming language and the SpaCy library in high school / I. Yu. Pikalov // MCU Journal of Informatics and Informatization of Education. 2024. № 2 (68). P. 48–61.

Введение

В настоящее время одной из ключевых компетенций специалиста любой сферы деятельности является знание современных технологий обработки большого объема информации и умение применять их в профессиональной деятельности. Это в первую очередь относится к неструктурированным текстовым данным, которые в настоящее время доступны в большом объеме и обработка которых без привлечения соответствующих инструментов и технологий не представляется возможной.

Если проанализировать федеральные государственные образовательные стандарты высшего образования магистратуры, то можно заметить, что в категории общепрофессиональных компетенций предусмотрена специальная группа — «Информационно-коммуникационные технологии для профессиональной деятельности». Это в полной мере относится к современным технологиям в области искусственного интеллекта и анализа больших текстовых данных [1].

К общим задачам анализа текстовой информации можно отнести выбор фактов, суммаризацию текстов, определение основных тем и рассматриваемых сущностей в больших объемах текстов. К специализированным задачам обработки текстов на естественном языке можно отнести анализ отзывов покупателей на различные товары, реакцию общества на происходящие процессы и принимаемые решения, существующий заказ на рынке труда и многое другое. В области искусственного интеллекта и анализа данных обозначенные вопросы относятся к задачам обработки естественного языка (*англ.* Natural Language Processing, NLP).

Natural Language Processing — особое направление искусственного интеллекта, которое занимается обработкой естественного языка. С использованием методов NLP решаются задачи суммаризации текстов, извлечения фактов об интересующем объекте, поиска ключевых слов, классификации текстов, определения тональности, продолжает развиваться машинный перевод, стремительно совершенствуются голосовые интерфейсы и чат-боты.

Исследователи А. Ю. Белякова и Ю. Д. Беляков в работе «Обзор задачи автоматической суммаризации текста» [2], помимо анализа данной задачи, приводят классификацию методов суммаризации и их сравнение, описывают основные методы подготовки текста. Эти методы необходимы при выполнении

любого проекта по анализу текстов. При изучении разделов учебных курсов, связанных с решением любых задач NLP, нужно рассматривать не только перечень подготовительных этапов и подходов к решению задач обработки текстов на естественном языке, но и обучать использованию самих методов и существующих инструментов.

В исследовании Н. С. Лагутиной, А. М. Васильева и Д. Д. Зафиевского «Задачи в области распознавания именованных сущностей» [3] подробно проанализирована задача распознавания именованных сущностей (*англ.* Named Entity Recognition, NER). Авторы предлагают обзор работ и проводят эксперименты с учетом тематических областей текстов. В качестве базовых технологий лидируют методы глубокого обучения. Выявлены основные проблемы задач распознавания сущностей, такие как дефицит эталонных наборов данных, высокие требования к вычислительным ресурсам и отсутствие анализа ошибок. Перспективным направлением исследований в области NER является развитие методов на основе обучения без учителя или на основе правил. Возможной базой предобработки текста для таких методов могут служить интенсивно развивающиеся модели языков в существующих инструментах NLP. И это подтверждает необходимость изучения инструментов NLP, использующих методы обработки, основанные на имеющихся языковых моделях.

В работе И. В. Левченко, Д. Б. Абушкина и П. А. Меренковой «Обработка естественного языка интеллектуальными системами» [4] предлагается методика преподавания модуля «Обработка естественного языка интеллектуальными системами» в общеобразовательном куске информатики. В качестве предметных результатов обучения выделено умение разрабатывать некоторые программные средства по тематике обработки естественного языка. При изучении обработки естественного языка в высшей школе это умение также должно быть сформировано и направлено на решение профессиональных задач.

В нашем исследовании предлагается один из возможных вариантов создания программного средства с использованием языка программирования Python и библиотеки SpaCy для решения задач обработки естественного языка.

Кроме разработки программных средств анализа текстов предлагаемая методика изучения методов и инструментов обработки естественного языка делает акцент на стадиях обработки текста, что способствует пониманию работы любых интеллектуальных систем по обработке естественного языка.

Применение конкретных методов обработки естественного языка зависит от решаемой задачи, но начальные этапы анализа текстов совпадают. Процесс обработки текстовых сообщений начинается с их загрузки, очистки и подготовки корпуса. Затем этот корпус кодируется любым типом представления текста, после чего следует его анализ и решаются конкретные поставленные задачи.

Методы исследования

В данной работе для обработки текстов на естественном языке будем использовать язык программирования Python и готовые функции библиотеки SpaCy, в качестве среды для выполнения программного кода воспользуемся Jupyter Notebook. Эти инструменты являются свободно распространяемыми, кросс-платформенными, простыми в использовании.

Jupyter Notebook — это среда разработки для написания и выполнения кода Python, веб-приложение с открытым исходным кодом, которое позволяет создавать документы, содержащие программный код, вывод полученных результатов, описательный текст, поддерживает средства навигации и визуализации. Среда разработки состоит из последовательности ячеек, каждая из которых содержит небольшой пример кода или документацию в формате Markdown. Разработчики и пользователи могут выполнить ячейку и сразу в документе увидеть результат ее работы под кодом. Это позволяет запускать программный код необходимыми фрагментами и вносить в него соответствующие изменения. Ячейки Jupyter Notebook также поддерживают аннотации с разметкой текста, аудиофайлы, видео, изображения, интерактивные диаграммы и многое другое. Описанные возможности программы позволяют использовать Jupyter Notebook при разработке проектов по анализу данных.

В данной работе описываются методы обработки текстов на естественном языке и приводится программный код с комментариями. Все методы с их описанием содержатся в файле проекта (файл с расширением ipynb), который предлагается студентам и слушателям, изучающим предлагаемый материал, а также исследователям, желающим использовать предлагаемый инструмент анализа текстов.

Более подробную информацию про среду Jupyter Notebook, включающую также и описание ее установки, можно получить на официальном сайте программы (https://jupyter.org/).

Последние разработки в области NLP доступны в открытых библиотеках Python, например в SpaCy, NLTK, TextaCy и NeuralCoref. Библиотека SpaCy — одна из самых популярных библиотек NLP наряду с NLTK. Основное различие между двумя библиотеками заключается в том, что NLTK содержит широкий спектр алгоритмов для решения одной проблемы, тогда как SpaCy содержит только один, но лучший алгоритм. Функционал библиотеки SpaCy позволяет решать широкий спектр задач.

В настоящей работе рассматривается анализ текстов на английском языке. Это обусловлено тем, что задача синтаксического анализа особенно хорошо решена для английского языка, где предложения имеют четкую структуру. Для первого знакомства с рассматриваемой технологией это методически оправданно, а в дальнейшем аналогичным образом можно решать задачи и для анализа текстов на другом языке.

Результаты исследования

Выделим основные этапы задач обработки естественного языка. Вначале будут следовать этапы предварительной обработки текстов. При этом следует учитывать, что тексты для анализа могут быть получены из социальных сетей, интернет-изданий, форумов и чатов и могут содержать различные последовательности символов, не несущих полезной информации.

При решении задач обработки текстов можно выделить следующие этапы:

1. Загрузка текста для анализа.
2. Очистка текста от символов, не несущих смысловой нагрузки.
3. Загрузка библиотек и создание модели текста.
4. Предварительный анализ исследуемого текста.
5. Решение конкретной прикладной задачи.

Изучим этапы задач обработки естественного языка и возможностей предлагаемых инструментов. Рассмотрим предлагаемый подход к изучению задач анализа текстов согласно выделенным этапам.

Выделим два варианта загрузки текста для анализа. Если текст имеет небольшой размер, например для знакомства с предлагаемыми методами, то достаточно просто присвоить его значение переменной.

```
# Загрузка текста для анализа путем копирования в переменную
text = """London is the capital and most populous city of England
and the United Kingdom. Standing on the River Thames in the south
east of the island of Great Britain, London has been a major settlement
for two millennia. It was founded by the Romans, who named
it Londinium."""
```

Второй вариант загрузки текста — это загрузка его из указанного файла. Например, если текст хранится в текстовом файле london.txt, то можно использовать следующую функцию:

```
# Загрузка текста для анализа из файла
import io
def open_file(name):
    f = io.open(name, encoding='utf-8')
    text = f.read()
    print(text[:100])
    return text

# Загрузка текста
text2 = open_file('london.txt')
```

Очистим текст от символов, не несущих смысловой нагрузки. Часто тексты для анализа содержат наборы символов, которые не несут смысловой нагрузки. Это особенно актуально для текстов, которые являются комментариями и содержат много знаков пунктуации, смайликов или других символов, выражающих эмоции. Для очистки текстов от лишних символов можно использовать

модуль re для создания регулярных выражений в Python. Приведем пример функции, которая из текста будет удалять символы русского алфавита, цифры и знаки «–» и «_»:

```
# Загрузка модуля re
import re

# Функция удаления ненужных символов
def removing_unnecessary(text):

# Создание регулярного выражения для удаления русских букв, цифр
и символов - и _
   reg = re.compile('[а-яёА-ЯЁ0-9-_]')
   text = reg.sub(' ', text)

# Создание регулярного выражения для замены нескольких пробельных
символов на пробел
   reg = re.compile('\s+')
   text = reg.sub(' ', text)

# Создание регулярного выражения для замены пробела и следующего
за ним знака препинания
   reg = re.compile(' [.!?]')
   text = reg.sub(' ', text)
   return text
```

Пример использования данной функции:

```
# Пример текста для очистки от лишних символов и слов
text3 = """“When a man is tired of London, he is tired of life; for there
is in London all that life can afford.” - это моя любимая цитата
про Лондон. --- 1000 раз прав____."""

# Очистка текста и вывод результата на экран
text_clear = removing_unnecessary(text3)
print(text_clear)
```

В результате мы получим очищенные от лишних слов и символов текст:

```
"When a man is tired of London, he is tired of life; for there is in London all that life
can afford."
```

Для использования библиотеки SpaCy первым шагом будет являться ее импорт (предварительно необходимо установить данную библиотеку) и загрузка необходимых языковых моделей. Для установки библиотеки достаточно выполнить команду:

```
# Установка библиотеки spacy (если она не установлена)
!pip install spacy
```

Установку библиотеки и языковой модели в среде достаточно выполнить 1 раз. Для английского языка существует 3 официальные модели, различающиеся

размером. Для демонстрации возможностей вполне достаточно установить небольшую модель en_core_web_sm.

```
# Установка языковой модели en_core_web_sm (если она не установлена)
!python -m spacy download en_core_web_sm
```

Пример кода с импортом библиотеки и загрузкой языковой модели для анализа английского языка en_core_web_sm:

```
# Импорт библиотеки spacy и загрузка языковой модели
import spacy
nlp = spacy.load('en_core_web_lg')
```

Для выполнения остальных операций с текстом нужно создать модель текста. Для этого используется команда nlp(), которая запускает целый конвейер по обработке текста. Для создания модели текста будут последовательно выполнены основные операции по его анализу. Такой конвейер по обработке текста называется пайплайном NLP, этапы конвейера можно настраивать под конкретную задачу.

Загруженный для обработки текст последовательно проходит через различные компоненты обработки и сохраняется как экземпляр объекта текстовой модели. Значение и результаты каждого этапа обработки можно узнать, изучив соответствующую документацию библиотеки. Мы применим заданный по умолчанию конвейер обработки.

На этапе предварительного анализа исследуемого текста рассмотрим некоторые примеры использования полученной модели текста для предварительного анализа текста.

Создадим модель текста, сохраненного в переменной text:

```
# Создание объекта модели текста
doc = nlp(text)
```

Разобьем текст на предложения. При анализе естественного языка важно правильно разделять текст на предложения, так как каждое предложение — это самостоятельная мысль или идея. Современные инструменты анализа текстов могут не просто разделять текст по определенным знакам препинания, а используют для этого более сложные методы, подходящие даже для работы с неформатированными фрагментами.

Для доступа к предложениям используется поле sents текстового объекта doc. Следующий пример кода создает список из предложений текста.

```
# Получение списка из предложений и вывод его на экран
list_sentence = [sentence for sentence in doc.sents]
for sentence in list_sentence:
  print(sentence)
```

Результат работы кода:

```
London is the capital and most populous city of England and the United Kingdom.
Standing on the River Thames in the south east of the island of Great Britain,
London has been a major settlement for two millennia.
It was founded by the Romans, who named it Londinium.
```

Далее идет токенизация — процесс разбиения текста на более мелкие части. Единицы разбиения можно выбирать свои.

Следующий пример кода создает список токенов (*англ.* token — знак, символ, жетон, талон; здесь — единица учета) и выводит его на экран.

```
# Получение списка токенов и вывод его на экран
token_list = [token for token in doc]
print(token_list)
```

Результат работы кода:

```
[London, is, the, capital, and, most, populous, city, of, England, and, the,
United, Kingdom, ., Standing, on, the, River, Thames, in, the, south, east,
of, the, island, of, Great, Britain, ,, London, has, been, a, major, settlement,
for, two, millennia, ., It, was, founded, by, the, Romans, ,, who, named,
it, Londinium, .]
```

Используя атрибут pos_ токена, можно посмотреть часть речи токена. Зная роль каждого слова в предложении, можно понять его общий смысл.

Для уменьшения выводимой информации некоторые действия будем выполнять только для первого предложения (первые 15 токенов). Создадим список токенов с указанием его принадлежности к части речи и выведем его на экран.

```
# Получение списка токенов и их частей речи
span_text = doc[:15]
token_list_pos = [f'{token} - {token.pos_}' for token in span_text]
print(token_list_pos)
```

Результат:

```
['London - PROPN', 'is - AUX', 'the - DET', 'capital - NOUN', 'and - CCONJ', 'most - ADV',
'populous - ADJ', 'city - NOUN', 'of - ADP', 'England - PROPN', 'and - CCONJ', 'the - DET',
'United - PROPN', 'Kingdom - PROPN', '. - PUNCT']
```

Для расшифровки названий частей речи можно воспользоваться функций explain:

```
# Расшифровка названий тегов
print(f"PROPN - {spacy.explain('PROPN')}")
print(f"AUX - {spacy.explain('AUX')}")
```

Результат:

```
PROPN — proper noun
AUX — auxiliary.
```

Следующий шаг — лемматизация, то есть процесс преобразования слова в его базовую форму.

Лемма (от *греч.* lemma — предложение, положение, доказанное утверждение) для каждого токена хранится в атрибуте lemma_.

Создание списка токенов и лемм и вывод его на экран:

```
# Получение списка токенов и лемм
token_lemma_list = [f"token: {token}, lemma: {token.lemma_}"
  for token in span_text]
  print(token_lemma_list)
```

Результат работы кода:

```
['token: London, lemma: London', 'token: is, lemma: be', 'token: the, lemma: the',
'token: capital, lemma: capital', 'token: and, lemma: and', 'token: most, lemma:
most', 'token: populous, lemma: populous', 'token: city, lemma: city', 'token: of,
lemma: of', 'token: England, lemma: England', 'token: and, lemma: and', 'token:
the, lemma: the', 'token: United, lemma: United', 'token: Kingdom, lemma:
Kingdom', 'token: ., lemma: .']
```

Важная часть работы — определение стоп-слов. Иногда при анализе текстовой информации приходится удалять слова, который часто употребляются, но не несут никакой смысловой нагрузки, так называемые стоп-слова. Библиотека SpaCy поставляется со списком стоп-слов по умолчанию. В английском языке очень много таких вспомогательных слов, например and, the, a.

Для обнаружения стоп-слов обычно используются готовые таблицы. Однако нет единого стандартного списка, подходящего в любой ситуации. Игнорируемые токены могут меняться, все зависит от особенностей проекта.

Иногда, наоборот, для оценки эмоциональной окраски текста такие слова не нужно удалять.

Для определения принадлежности токена к списку стоп-слов используется атрибут токена is_stop. Создадим список токенов без стоп-слов:

```
# Создание списка токенов без стоп-слов
filtered_tokens_list = [token for token in doc if not token.is_stop]
print(filtered_tokens_list)
```

Результат:

```
[London, capital, populous, city, England, United, Kingdom, ., Standing, River,
Thames, south, east, island, Great, Britain, ,, London, major, settlement,
millennia, ., founded, Romans, ,, named, Londinium, .]
```

Можно получить множество из стоп-слов, которые были в тексте:

```
# Получение множества стоп-слов, которые содержатся в тексте
  stop_words_text = set([token.text.lower() for token in doc
if token.is_stop])
  print(stop_words_text)
```

Результат:

{'of', 'most', 'the', 'who', 'a', 'is', 'been', 'on', 'in', 'for', 'it', 'was', 'by', 'two', 'has', 'and'}

При необходимости список стоп-слов можно менять.

Далее выполняем синтаксический анализ зависимостей, под которым будем понимать установление взаимосвязи между словами в предложении. Иначе его называют парсингом зависимостей. Конечная цель этого шага — построение дерева, в котором каждый токен имеет единственного родителя. Корнем может быть главный глагол.

Используя атрибут dep_ токена, можно посмотреть синтаксическую зависимостей, то есть отношение между токенами. Используя атрибут head токена, можно посмотреть токен, от которого он зависит.

Выведем таблицу синтаксической зависимости токенов, входящих в первое предложение, их части речи, синтаксическую зависимость и токены, от которого они зависят:

```
# Вывод токена, его части речи, синтаксической зависимости и токена, от которого
он зависит
for token in span_text:
    token_text = token.text
    token_pos = token.pos_
    token_dep = token.dep_
    token_head = token.head.text
    print(f"{token_text:<12}{token_pos:<10}" \
          f"{token_dep:<10}{token_head:<12}")
```

Результат показан на рисунке 1.

```
London      PROPN     nsubj     is
is          AUX       ROOT      is
the         DET       det       capital
capital     NOUN      attr      is
and         CCONJ     cc        capital
most        ADV       advmod    populous
populous    ADJ       amod      city
city        NOUN      conj      capital
of          ADP       prep      city
England     PROPN     pobj      of
and         CCONJ     cc        England
the         DET       det       Kingdom
United      PROPN     compound  Kingdom
Kingdom     PROPN     conj      England
.           PUNCT     punct     is
```

Рис. 1. Таблица синтаксической зависимости токенов

Для расшифровки терминов можно воспользоваться функций explain (уже использовали при расшифровке названий частей речи).

Для отображения зависимости токенов можно построить синтаксическое дерево зависимостей (рис. 2). Для этого используется визуализатор displaCy, которому нужно просто передать строку для построения дерева.

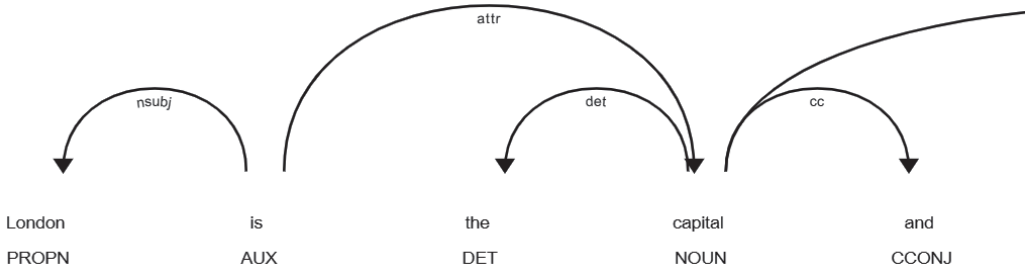


Рис. 2. Фрагмент нарисованного синтаксического дерева зависимостей

```
# Загрузка визуализатора displacy
from spacy import displacy

# Рисование синтаксического дерева зависимостей
displacy.render(span_text, style='dep', jupyter=True)
```

Многие английские предложения неоднозначны и сложны для анализа. В таких случаях модель делает предположение о наиболее вероятном значении, хотя это не всегда получается.

Дальнейший шаг — получение именованных сущностей. Именованные сущности содержатся в атрибуте ents объекта nlp. Для получения метки для сущности используется атрибут label_:

```
# Вывод именованных сущностей, их меток и расшифровок меток
for entity in doc.ents:
  print(f"{entity.text} ({entity.label_}: {str(spacy.explain(entity.label_))})")
```

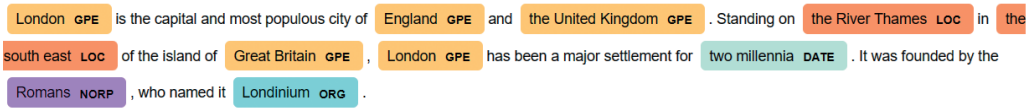
Результат:

```
London (GPE: Countries, cities, states)
England (GPE: Countries, cities, states)
the United Kingdom (GPE: Countries, cities, states)
the south east (LOC: Non-GPE locations, mountain ranges, bodies of water)
Great Britain (GPE: Countries, cities, states)
London (GPE: Countries, cities, states)
two millennia (DATE: Absolute or relative dates or periods)
Romans (NORP: Nationalities or religious or political groups)
Londinium (ORG: Companies, agencies, institutions, etc.)
```

Визуализатор displaCy может наглядно обозначить сущности прямо в тексте:

```
# Вывод документа с выделенными сущностями
displacy.render(doc, style='ent', jupyter=True)
```

Результат:



Выполним шаг «получение существительных». Существительные содержатся в атрибуте noun_chunks объекта nlp. Для получения списка существительных воспользуемся следующим кодом:

```
# Получим список существительных
list_noun = [noun for noun in doc.noun_chunks]
print(list_noun)
```

Результат:

```
[London, the capital, most populous city, England, the United Kingdom,
the River Thames, the south east, the island, Great Britain, London, a major
settlement, two millennia, It, the Romans, who, it, Londinium]
```

Мы рассмотрели наиболее часто используемые атрибуты токенов, в документации можно познакомиться и с остальными. К токену можно обращаться по номеру и узнавать значения и свойства его атрибутов.

Обратимся к примерам решения задач NLP. Рассмотрим задачу по извлечению фактов из текста. Если текст имеет большой размер, то, для того чтобы понять его основную мысль, бывает достаточно посмотреть автоматически выбранные из него фактические сведения, под которыми мы будем понимать связку заданной сущности с глагольной леммой, с которой ассоциируется сущность. Для решения данной задачи подходит библиотека TextaCy (https://pypi.org/project/textacy/).

Пример кода, который выполняет извлечение фактов из текста на примере поиска сочетаний сущности «London» с различными формами глагола «bi»:

```
# Текст для анализа
text = """London is the capital and most populous city of England
and the United Kingdom. Standing on the River Thames in the south east
of the island of Great Britain, London has been a major settlement
 for two millennia.  It was founded by the Romans, who named it
Londinium.
"""

# Установка библиотеки textacy (если она не установлена)
# !pip install textacy

# Импорт библиотеки textacy.extract
import textacy.extract

# Извлечение выражений со словом London
statements=textacy.extract.semistructured_statements(doc,
```

```
entity="London", cue='be')

# Вывод результатов
print("Here are the things I know about London:")
for statement in statements:
    subject, verb, fact = statement
    print(f" - {' '.join(map(str, fact))}")
```

Результат:

```
Here are the things I know about London:
 - the capital and most populous city of England and the United Kingdom
 - a major settlement for two millennia
```

Среди задач, с решением которых можно познакомить в рамках изучения данного раздела, можно выделить нахождение часто упоминаемых фрагментов и частотный анализ слов.

Заключение

В настоящей работе обосновывается необходимость изучения методов обработки естественного языка в высшей школе. Предложенный в исследовании подход к изучению обработки текстовых данных имеет следующие преимущества:

1) в нем используются свободно распространяемые программные средства и библиотеки;

2) при рассмотрении материала используется готовый программный код для анализа текстов, что позволяет применять его не только людьми, имеющими навыки программирования, но и специалистами и обучающимися из любой предметной области. Это также делает возможным включать предлагаемый раздел в любые учебные дисциплины, где нужно формировать соответствующие компетенции;

3) изучаются современные инструменты анализа данных, используемые специалистами при реализации проектов по анализу и визуализации больших данных, которые в дальнейшем могут применяться в профессиональной деятельности;

4) рассматриваются все необходимые этапы решения задач обработки естественного языка с демонстрацией их реализации с учетом выбранного инструментария, что соответствует принципу фундаментализации содержания образования и способствует пониманию принципов работы любой интеллектуальной системы.

Список источников

1. Гриншкун В. В. Развитие образования в эпоху четвертой промышленной революции / В. В. Гриншкун, Г. А. Краснова // Информатика и образование. 2017. № 1 (280). С. 42–45.

2. Белякова А. Ю. Обзор задачи автоматической суммаризации текста / А. Ю. Белякова, Ю. Д. Беляков // Инженерный вестник Дона. 2020. № 10 (70). С. 142–159.

3. Лагутина Н. С. Задачи в области распознавания именованных сущностей: технологии и инструменты / Н. С. Лагутина, А. М. Васильев, Д. Д. Зафиевский // Моделирование и анализ информационных систем. 2023. Т. 30, № 1. С. 64–85.

4. Левченко И. В. Модуль «Обработка естественного языка интеллектуальными системами» в общеобразовательном курсе информатики / И. В. Левченко, Д. Б. Абушкин, П. А. Меренкова // Вестник МГПУ. Серия «Информатика и информатизация образования». 2021. № 1 (55). С. 30–42.

References

1. Grinshkun V. V. The development of education in the era of the Fourth Industrial Revolution / V. V. Grinshkun, G. A. Krasnova // Informatics and education. 2017. № 1 (280). P. 42–45.

2. Belyakova A. Yu. Review of the problem of automatic text summarization / A. Yu. Belyakova, Yu. D. Belyakov // Engineering Bulletin of the Don. 2020. № 10 (70). P. 142–159.

3. Lagutina N. S. Tasks in the field of named entity recognition: technologies and tools / N. S. Lagutina, A. M. Vasiliev, D. D. Zafievsky // Modeling and analysis of information systems. 2023. Vol. 30, № 1. P. 64–85.

4. Levchenko I. V. Module “Natural language processing by intelligent systems” in the general education course of computer science / I. V. Levchenko, D. B. Abushkin, P. A. Merenkova // MCU Journal of Informatics and Informatization of Education. 2021. № 1 (55). P. 30–42.

Статья поступила в редакцию: 22.01.2024;
одобрена после рецензирования: 22.03.2024;
принята к публикации: 22.03.2024.

The article was submitted: 22.01.2024;
approved after reviewing: 22.03.2024;
accepted for publication: 22.03.2024.

Информация об авторе / Information about author:

Пикалов Иван Юрьевич — кандидат педагогических наук, доцент, доцент кафедры компьютерных технологий и информатизации образования, руководитель научно-методического центра разработки информационных систем и анализа данных, Курский государственный университет, Курск, Россия.

Ivan Yu. Pikalov — Candidate of Pedagogical Sciences, Associate Professor, Associate Professor of the Department of Computer Technology and Informatization of Education, Head of the Center for Information Systems Development and Data Analysis, Kursk State University, Kursk, Russia.

pikalov@kursksu.ru