Загружаем необходимые библиотеки

```
Ввод [83]: import numpy as np
           import warnings
           warnings.filterwarnings('ignore')
           warnings.warn('DelftStack')
Ввод [84]: import matplotlib.pyplot as plt
           import pandas as pd
           import time
           from pathlib import Path
           import csv
           import openpyxl
           import nltk
           from nltk.probability import FreqDist
           from nltk.corpus import stopwords
           from nltk.tokenize import word tokenize
           nltk.download('punkt')
           nltk.download('stopwords')
           #!pip install ipywidgets
           [nltk_data] Downloading package punkt to
           [nltk_data]
                           C:\Users\lapte\AppData\Roaming\nltk_data...
           [nltk_data]
                         Package punkt is already up-to-date!
           [nltk_data] Downloading package stopwords to
           [nltk_data]
                           C:\Users\lapte\AppData\Roaming\nltk_data...
           [nltk_data]
                         Package stopwords is already up-to-date!
 Out[84]: True
Ввод [29]:
           Enabling notebook extension jupyter-js-widgets/extension...
                 - Validating: ok
```

Стр. 1 из 14 26.11.2023, 22:31

Количество ядер процессора

для каждой машины свои значения

```
Ввод [3]: import multiprocessing as mp
n_workers = mp.cpu_count()

4 workers are available
```

Загрузим файл и посмотрим на него

Мультипроцессорная обработка данных: Посчитаем количество строк в файле с помощью пандас

```
Ввод [92]: # Вариант с циклом
dataset_path =Path('parsed_data.csv')
chunksize = 100
df_chunks = pd.read_csv(
    dataset_path,
    encoding='utf-8',
    chunksize=chunksize)

start_time = time.time()
sizes = []
for chunk in df_chunks:
    sizes.append(len(chunk))
print(f'Количество строк {sum(sizes)}')
print(f'Время выполнения: {round(time.time() - start_time, 2)} секунд.')

Количество строк 11375
```

Количество строк 11375 Время выполнения: 3.05 секунд.

Выведем в консоли размер полученного объекта, названия его столбцов и продолжительность операции чтения файла.

Стр. 2 из 14 26.11.2023, 22:31

```
Ввод [93]: %%time
              df = pd.read_csv('parsed_data.csv', encoding='utf-8')
              Shape: (11375, 2)
              Column Names:
              Index(['url', 'content'], dtype='object')
              CPU times: total: 2.53 s
              Wall time: 2.82 s
Ввод [94]:
  Out[94]:
                                                                               content
               0 https://0-100km.ru
                                                                                  NaN
                   https://0-chan.ru
                                      анонимная имиджборда Лого нультиреча вкл луп в...
               2
                     https://0-12.ru
                                                                                  NaN
               3
                      https://0--5.ru Видионаблюдение Монтаж ремонт обслуживание Вид...
                     https://0-0-1.ru
                                      Главная Корпоративный сайт по продаже компьюте...
                   https://0-base.ru
                                       Без названия Без названия Без названия Поиск л...
               6
                    https://0-nds.ru
                                      Закладки Мед в Заволжск Закладки Мед в Заволжс...
               7
                    https://0-ede.ru
                                      О еде Рецепты блюд и секреты кулинарии Рецепты...
               8
                    https://0--0-0.ru
                                      Автомобильный портал автоновости отзывы автовл...
               9
                      https://0-d.ru
                                      Приветствуем Сайт только что создан Содержимое...
```

Удалим строчки содержащие NaN

Стр. 3 из 14 26.11.2023, 22:31

Функция возвращяющая dataframe с некорректными символами содержащимися в столбце

Стр. 4 из 14 26.11.2023, 22:31

```
Ввод [96]: def get_illegal_characters(df, column_name, legitimate_characters, encoding='utf8', DEBUG=False):
             # получим все не легитимные символы из датафрейма
             result = df['content'].str.extractall(legitimate characters)
             # Решим проблему с повторениями.
             result = result[0].unique()
             result = pd.DataFrame(result)
             # Добавим столбец с нех кодировкой символов.
             result['hex_code'] = result[0].str.encode(encoding)
             # Дадим адекватные имена столбцам.
             result.rename(columns={0: 'character'}, inplace=True)
             if (DEBUG):
               print("Найдено", result.shape[0], "уникальных символов не подходящих под заданный шаблон.")
               print("Найдены следующие символы:\n", result)
             return result
           # Шаблон "легитимных символов". Использование этих символов допустимо в русском языке.
           # Естественно символ пробела мы тоже оставим, иначе наши слова "склеятся".
           legitimate_characters = "([^a-яёА-ЯЁ])"
           # Имя столбца, который необходимо проверить
           column name = 'content'
           bad_symbols = get_illegal_characters(df, column_name, legitimate_characters)
           bad_symbols
 Out[96]:
              character hex_code
```

Создаем функцию, которая очищает текст от ненужных символов

Стр. 5 из 14 26.11.2023, 22:31

```
BBOД [97]: import re

def clean_text(text):
    template = '|'.join(map(re.escape, bad_symbols['character']))
    # Удаление ненужных символов из строки.
    text = re.sub(template, '', text, flags=re.UNICODE)
    return text
```

Протестируем различные методы очистки слов в тексте

Последовательный метод очистки текста

```
Ввод [10]: #!pip install --upgrade jupyter
           Requirement already satisfied: jupyter in c:\users\lapte\anaconda3\envs\nlp pyt3 10 13\lib\site-packages (1.
           0.0)
           Requirement already satisfied: notebook in c:\users\lapte\anaconda3\envs\nlp pyt3 10 13\lib\site-packages (f
           rom jupyter) (6.5.4)
           Requirement already satisfied: qtconsole in c:\users\lapte\anaconda3\envs\nlp pyt3 10 13\lib\site-packages
           (from jupyter) (5.4.4)
           Requirement already satisfied: jupyter-console in c:\users\lapte\anaconda3\envs\nlp pyt3 10 13\lib\site-pack
           ages (from jupyter) (6.6.3)
           Requirement already satisfied: nbconvert in c:\users\lapte\anaconda3\envs\nlp pyt3 10 13\lib\site-packages
           (from jupyter) (6.5.4)
           Requirement already satisfied: ipykernel in c:\users\lapte\anaconda3\envs\nlp pyt3 10 13\lib\site-packages
           (from jupyter) (6.25.0)
           Requirement already satisfied: ipywidgets in c:\users\lapte\anaconda3\envs\nlp pyt3 10 13\lib\site-packages
           (from jupyter) (8.1.1)
           Requirement already satisfied: comm>=0.1.1 in c:\users\lapte\anaconda3\envs\nlp pyt3 10 13\lib\site-packages
           (from ipykernel->jupyter) (0.1.4)
           Requirement already satisfied: debugpy>=1.6.5 in c:\users\lapte\anaconda3\envs\nlp pyt3 10 13\lib\site-packa
           ges (from ipykernel->jupyter) (1.6.7)
           Requirement already satisfied: ipython>=7.23.1 in c:\users\lapte\anaconda3\envs\nlp pyt3 10 13\lib\site-pack
```

Стр. 6 из 14 26.11.2023, 22:31

```
Ввод [98]: from tqdm import tqdm
           from time import sleep
           tqdm.pandas()
           %time df['content'] = df['content'].progress_apply(clean_text)
           sleep(0.01)
                            9919/9919 [00:25<00:00, 393.67it/s]
           CPU times: total: 24.6 s
           Wall time: 25.2 s
           Очищаем текст с помощью модуля multiprocessing: не работает в Anaconda
Ввод [ ]: %time
           import multiprocessing as mp
           from tqdm.notebook import tqdm
           n_{workers} = 4 # количество распараллеленых процессов(воркеров)
           # Создание многопроцессорного пула
           p = mp.Pool(n_workers)
           # Примените функцию cleana_text к каждому элементу столбца 'content'
           # путем передачи отдельных строк
           text = list(tqdm(p.imap(clean_text, df['content']), total=len(df)))
           p.close()
           p.join()
           CPU times: total: 0 ns
           Wall time: 0 ns
             0%|
                           | 0/9919 [00:00<?, ?it/s]
```

Анализ текста

Стр. 7 из 14 26.11.2023, 22:31

```
Ввод [99]: from collections import Counter

text = df['content'].tolist()

# Функция возвращающая общее количество уникальных слов и общее кол-во их вместе взятая def word_stats(word_counts):
    num_unique = len(word_counts)
    counts = list(word_counts.values())
    return num_unique, counts

# Вычисление частот слов с помощью счетчика word_counts = Counter(text)

# вызов функции
    num_unique, counts = word_stats(word_counts)
    print("Число уникальных слов:", num_unique)
    print("сумма частот уникальных слов:", sum(counts))

Число уникальных слов: 8991
    сумма частот уникальных слов: 9919
```

Популярные слова

```
Ввод [100]: #coxpaнum данные столбца 'content' с типом str, чтобы можно было работать с nltk content = ' '.join(df['content']) # Объединить с пробелом в качестве разделителя text=content.lower() # все с маленькой буквы

print(type(text))

<class 'str'>
```

анонимная имиджборда лого нультиреча вкл луп выкл луп лого нультиреча вкл луп нас не работает главная все до ски тематика аниме радиоэлектроника мех политач тульпафорсинг и саморазвитие зеркало зеркало не работает гла вная метадоска с лучшим обсуждение медиаконтента такого как книги комиксы фильмы ани

Стр. 8 из 14 26.11.2023, 22:31

```
Ввод [13]: %%time
           start_time = time.time()
           # токенизируем текст
           words = word_tokenize(text)
           # Создаем список со стоп словами для русского языка
           stop_words = stopwords.words('russian')
           stop_words.extend(['это', 'что', 'всё', 'м', 'г', 'p'])
           # Удаляем стоп слова
           nouns = [word for word in words if word not in stop_words]
           # Вычисление частот слов
           fdist = FreqDist(nouns)
           # Получите 10 популярных слов
           most_common_words = fdist.most_common(10)
           print(most_common_words)
           [('отдых', 97377), ('руб', 34617), ('сайт', 24983), ('работа', 23591), ('купить', 23536), ('данных', 19696),
           ('подробнее', 19681), ('поиск', 17285), ('ремонт', 17136), ('доставка', 15209)]
           CPU times: total: 2min 6s
           Wall time: 2min 8s
```

Стр. 9 из 14 26.11.2023, 22:31

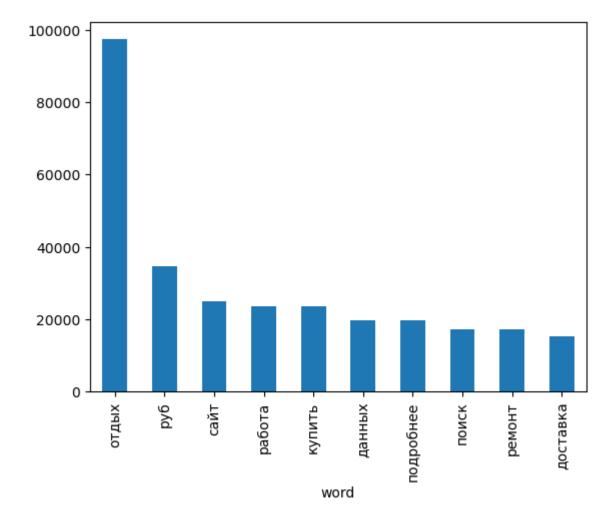
```
Ввод [18]: #преобразуем кортеж в таблицу для лучшей визуализации df2 = pd.DataFrame(most_common_words, columns=['word', 'count'])
```

| Out[18]: | | word | count |
|----------|---|-----------|-------|
| | 0 | отдых | 97377 |
| | 1 | руб | 34617 |
| | 2 | сайт | 24983 |
| | 3 | работа | 23591 |
| | 4 | купить | 23536 |
| | 5 | данных | 19696 |
| | 6 | подробнее | 19681 |
| | 7 | поиск | 17285 |
| | 8 | ремонт | 17136 |
| | 9 | доставка | 15209 |

Нарисуем гистограмму распределения часто встречющихся слов

Стр. 10 из 14 26.11.2023, 22:31





Визуализация популярности слов в виде облака

Скачиваем необходимые библиотеки

Ввод [16]: from wordcloud import WordCloud import matplotlib.pyplot as plt %matplotlib inline

Стр. 11 из 14 26.11.2023, 22:31

Выведим слова в виде красивой картинки, используя библиотеку WordCloud

```
Ввод [17]: wordcloud = WordCloud(width=800, height=400, background_color='white').generate(text)

plt.figure(figsize=(10, 5))
plt.imshow(wordcloud, interpolation='bilinear')
plt.axis("off")
plt.show()
```



Частота вхождения заданных слов в столбце

Стр. 12 из 14 26.11.2023, 22:31

```
Ввод [79]: print(f"количество сайтов, содержащих слово Контакты: {df.content.str.contains('контакты', na=False).sum()}\n
           print(f"количество сайтов, содержащих слово ИНН: {df.content.str.contains('ИНН', na=False).sum()}\n")
           print(f"количество сайтов, содержащих слово телефон: {df.content.str.contains('Телефон', na=False).sum()}\n")
           print(f"количество сайтов, содержащих слово адрес: {df.content.str.contains('адрес', na=False).sum()}\n")
           print(f"количество сайтов, содержащих слово МФТИ: {df.content.str.contains('МФТИ', na=False).sum()}\n")
           df1 = df[df['content'].str.contains("ΜΦΤΝ")]
           количество сайтов, содержащих слово Контакты: 889
           количество сайтов, содержащих слово ИНН: 810
           количество сайтов, содержащих слово телефон: 2773
           количество сайтов, содержащих слово адрес: 2571
           количество сайтов, содержащих слово МФТИ: 3
                                  url
                                                                                  content \
           1255 https://1-variant.ru (https://1-variant.ru) Главная Вариант это дефолтный вариант Вариант ...
           5655
                  https://lcov-edu.ru (https://lcov-edu.ru) Методы решения физико математических задач выс...
           9951
                  https://231-tech.ru (https://231-tech.ru) Главная Главная страница сайта студенческих се...
                 query_count
           1255
                           1
           5655
                           1
           9951
                           1
```

Частота вхождения заданных словосочетаний

Стр. 13 из 14 26.11.2023, 22:31

```
Ввод [78]: \# Ищем словосочетание в столбце 'content'
           result = df[df['content'].str.contains('адрес челябинск', case=False, na=False)]
           print(result)
           #Запись результата в excel, если небходимо
                                       url \
                         https://007spa.ru (https://007spa.ru)
           150
                          https://127ds.ru (https://127ds.ru)
           3671
           4115
                      https://147school.ru (https://147school.ru)
                     https://lccrm-tech.ru (https://lccrm-tech.ru)
           5518
           7386
                        https://lsignal.ru (https://lsignal.ru)
           9630
                        https://22-volt.ru (https://22-volt.ru)
           9717
                        https://2221001.ru (https://2221001.ru)
           10734
                  https://24facecontrol.ru (https://24facecontrol.ru)
                                                             content query_count
           150
                  Спа салон для мужчин Эротического массажа в Че...
                                                                                1
                  МБДОУ Детский сад г Челябинска Муниципальное б...
           3671
                                                                                1
                  Школа Челябинска официальный сайт МАОУ СОШ Чел...
           4115
                                                                                1
           5518
                  С решения для взаимоотношения с клиентами Глав...
                                                                                1
           7386
                  Автошкола в Троицке Сигнал Автошкола Сигнал Ав...
                                                                                1
           9630
                  Газклимат газовые котлы и газовое оборудование...
                                                                                1
           9717
                  АвтоАдвокат Услуги автоадвоката в Челябинске п...
                                                                                1
           10734 ЛайтШоп Главная Интернет магазин заказать купи...
```

Стр. 14 из 14 26.11.2023, 22:31

```
<!DOCTYPE html>
<html lang="ru">
<head>
    <meta charset="UTF-8" />
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Главная</title>
    <link rel="stylesheet" href="style.css">
</head>
<body>
    <header>
        <div class="center">
            <div class="logo">
href="https://github.com/BoginskiyVV/webpageDiploma">Github</a>
href="https://drive.google.com/drive/folders/1ZugqcHTbfqaNu64ULXxcGxFxfEIRDbx
z">GoogleDrive</a>
            </div>
            <div class="menu">
                <a href="contents.html">Содержание</a>
                <a href="introduction.html">Введение</a>
                <a href="chapter1.html">Глава I</a>
                <a href="chapter2.html">Глава II</a>
                <a href="conclusion.html">Заключение</a>
                <a href="appendix.html">Приложения</a>
                <a href="literature.html">Список литературы</a>
                <!-- <a
href="https://github.com/BoginskiyVV/webpageDiploma">Github</a> -->
                <!-- <a
href="https://drive.google.com/drive/folders/1ZugqcHTbfqaNu64ULXxcGxFxfEIRDbx
z">GoogleDrive</a> -->
            </div>
        </div>
        <!--Center-->
    </header>
    <section class="main">
        <div class="center">
            <div class="main cta">
                <h2>Коллективный дипломный проект на тему:
```

```
«Применение наиболее значимых компонентов освоенного в
процессе
                   обучения технологического стека для разработки
                   поисково-аналитического комплекса»</h2>
                Исполнители:<br>
                   Афанасьева А.Ю., ВІ аналитик, группа 5295<br>
                   Богинский В.В., Аналитик больших данных, группа 5321<br>
                   Евстифеева Ю.В., ВІ аналитик, группа 5295<br>
                   Лаптева Ф.Р., Аналитик больших данных, группа 5321<br>
                   Майбродский М.В., Аналитик больших данных, группа
5321<br>
                   Пиев А.Б., Аналитик больших данных, группа 5321
            </div>
            <div class="diferenciais">
                <div class="diferencial-single" id="single-1">
                   <img src="src/Anna.jpg"><br>
                   <h2>Афанасьева А.Ю.</h2>
                   ВІ аналитик, группа 5295
               </div>
                <div class="diferencial-single" id="single-2">
                   <img src="src/Vladimir.jpg"><br>
                   <h2>Богинский В.В.</h2>
                   <р>Аналитик больших данных, группа 5321</р>
                </div>
                <div class="diferencial-single" id="single-3">
                   <img src="src/Uliya.jpg"><br>
                   <h2>Евстифеева Ю.В.</h2>
                   ВІ аналитик, группа 5295
               </div>
           </div>
       </div>
    </section>
    <section class="main">
       <div class="center">
           <div class="diferenciais">
                <div class="diferencial-single" id="single-1">
                   <img src="src/Firyuza.jpg"><br>
                   <h2>Лаптева Ф.Р.</h2>
                   <р>Аналитик больших данных, группа 5321</р>
                <div class="diferencial-single" id="single-2">
                   <img src="src/Maxim.jpg"><br>
                   <h2>Майбродский М.В.</h2>
                   <р>Аналитик больших данных, группа 5321</р>
```

```
</div>
                <div class="diferencial-single" id="single-3">
                    <img src="src/Andrew.jpg"><br>
                    <h2>Пиев А.Б.</h2>
                    <р>Аналитик больших данных, группа 5321</р>
            </div>
        </div>
    </section>
</body>
</html>
<head>
    <meta charset="UTF-8" />
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Содержание</title>
    <link rel="stylesheet" href="style.css">
</head>
<body>
    <header>
        <div class="center">
            <div class="logo">
                <a href="index.html">На главную</a>
                <!-- <a
href="https://drive.google.com/drive/folders/1ZugqcHTbfqaNu64ULXxcGxFxfEIRDbx
z">GoogleDrive</a> -->
            </div>
            <div class="menu">
                <a href="introduction.html">Введение</a>
                <a href="chapter1.html">Глава I</a>
                <a href="chapter2.html">Глава II</a>
                <a href="conclusion.html">Заключение</a>
                <a href="appendix.html">Приложения</a>
                <a href="literature.html">Список литературы</a>
                <!-- <a
href="https://github.com/BoginskiyVV/webpageDiploma">Репозиторий Github</a>
                  <a class="btn-menu" href="#">Ссылка</a> -->
            </div>
        </div>
        <!--Center-->
```

```
<section class="main">
        <div class="center">
            <div class="main cta">
                <h2>Содержание</h2>
                <br>
                    1. Введение. <br>
                    <br>
                    2. ГЛАВА I. Теоритическая часть. <br>
                    2.1. Язык Python для поиска информации в сети Интернет и
парсинга сайтов. <br>
                    2.1.1. Общая информация по языку Python и парсингу
сайтов. <br>
                    2.1.2. Основы использования библиотеки requests. <br>
                    2.1.3. Основы использования библиотеки BeautifulSoup. <br>
                    2.1.4. Основы использования библиотеки Scrapy. <br>
                    2.1.5. Основы использования библиотеки Selenium. <br>
                    2.2. Средства хранения данных при парсинге сайтов. <br>
                    2.2.1. Использование формата CSV при парсинге сайтов. <br>
                    2.2.2. Использование формата JSON при парсинге
caŭtos. <br>
                    2.2.3. Использование SQL при парсинге сайтов. <br>
                    2.2.4. Использование Hadoop при парсинге сайтов. <br>
                    2.3. Использование языка Python для анализа данных. <br>
                    2.3.1. Общая информация по возможностям языка Python для
анализа данных. <br>
                    2.3.2. Основы использования библиотеки Pandas для анализа
данных.<br>
                    2.3.3. Основы использования библиотек Matplotlib и
Seaborn для анализа данных. <br>
                    2.4. Использование SQL для анализа данных. <br>
                    2.5. Использование BI инструментов для анализа данных и
вывода результатов. <br>
                    2.6. dfsgdfg
                    <br>
                    3. ГЛАВА II. Разработка и тестирование поисково-
аналитического комплекса. <br>
                    3.1. Парсер на языке Python с загрузкой данных в SQL
формат.<br>
                    3.2. Реализация на Python многопоточной обработки данных
для парсинга сайтов
                    с последующим анализом в Jupiter notebook. <br>
                    3.3. Анализ полученных данных средствами SQL. <br>
```

</header>

```
3.4. Анализ и вывод полученных данных с помощью
инструментов BI.<br>
                    3.5. Поисково-аналитический Telegram bot на Python. <br>
                    3.6. Вывод результатов проекта с помощью создания сайта
на HTML/CSS.<br>
                    <br>
                    4. Заключение.<br>
                    <br>
                    5. Приложения. <br>
                    <br>
                    6. Список литературы. <br>
                </div>
        </div>
        </div>
    </section>
</body>
</html>
<head>
    <meta charset="UTF-8"/>
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Введение</title>
    <link rel="stylesheet" href="style.css">
</head>
<body>
    <header>
        <div class="center">
            <div class="logo">
                <a href="index.html">На главную</a>
                <!-- <a
href="https://drive.google.com/drive/folders/1ZugqcHTbfqaNu64ULXxcGxFxfEIRDbx
z">GoogleDrive</a> -->
            </div>
            <div class="menu">
                  <a href="contents.html">Содержание</a>
                  <a href="chapter1.html">Глава I</a>
                  <a href="chapter2.html">Глава II</a>
                  <a href="conclusion.html">Заключение</a>
                  <a href="appendix.html">Приложения</a>
                  <a href="literature.html">Список литературы</a>
```

```
<!-- <a
href="https://github.com/BoginskiyVV/webpageDiploma">Репозиторий Github</a>
                <a class="btn-menu" href="#">Ссылка</a> -->
              </div>
       </div>
       <!--Center-->
   </header>
   <section class="main">
       <div class="center">
           <div class="main cta">
               <h2>Введение</h2>
                   <b>1. Введение.</b><br>
                  <br>
                     В настоящее время сеть Интернет
содержит огромное
                  количество информации, и задача по её поиску, обработке и
анализу становится
                  все более актуальной. Для разработки эффективных решений
задач такого рода
                  необходимо использовать соответствующие современные
технологические
                  инструменты. В данном дипломном проекте рассматривается
применение наиболее
                  значимых компонентов, освоенных в процессе обучения, для
разработки
                  поисково-аналитического комплекса. <br>
                     В первой главе рассматриваются
основы использования
                  языка программирования Python для поиска информации в
сети Интернет и парсинга
                  данных с веб-сайтов. Эта часть дипломного проекта
включает в себя общую информацию
                  по языку Python и его возможностям, основы использования
библиотеки requests, основы
                  использования библиотеки BeautifulSoup для парсинга HTML,
основы использования
                  библиотеки Scrapy, а также основы использования
библиотеки Selenium. <br>
                     Также в этой главе
рассматривается использование различных
                  походов для хранения анных, в том числе таких, как
использование файлов CSV и JSON,
```

возможности взаимодействия Python c SQL, а также средства хранения и обработки больших

данных с помощью Hadoop.

Дополнительно проводится обзор возможностей языка Python

для анализа данных, включая основы работы с библиотекой Pandas для анализа данных

и основы использования библиотек Matplotlib и Seaborn для визуализации данных. Также

рассматривается использование SQL для анализа данных и использование BI инструментов

для анализа данных и вывода результатов. Завершается глава обсуждением использования

стека HTML/CSS для представления результатов проделанной аналитической работы.

Bo второй главе описывается процесс разработки и тестирования

командой проекта поисково-аналитического комплекса. Разработка поискового механизма

производится на языке Python, команда осуществляет выбор и настройку соответствующих

инструментов для поиска и парсинга данных. Также в практической плоскости реализуется

возможность использования файлов CSV и использование средств SQL для хранения полученных

данных, производится анализ полученных данных с помощью языков Python и SQL, а также

анализ и вывод полученных данных с помощью доступных инструментов ВІ. Дополнительно

команда представляет использование стека HTML/CSS для отображения результатов выполнения

дипломной работы в качестве веб-страницы.

B заключении подводятся итоги проведенной работы, резюмируются

результаты и достижения проекта. Оцениваются его преимущества и недостатки, а также

даются рекомендации по его дальнейшему развитию и применению. В приложениях представлены

дополнительные материалы, которые могут быть полезны для более подробного изучения

темы проекта, в том числе исходный код разработанных инструментов, дополнительные схемы

и диаграммы, а также результаты дополнительных экспериментов и исследований.


```
  В конце работы приводится список
использованной литературы и
                    других источников информации, которые были использованы
при разработке проекта.
                    В список входят книги, онлайн-ресурсы и другие источники,
которые помогли в освоении
                    и применении технологического стека для разработки
поисково-аналитического комплекса. <br>
                </div>
            </div>
        </div>
   </section>
</body>
</html>
<head>
    <meta charset="UTF-8" />
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Глава 1</title>
    <link rel="stylesheet" href="style.css">
</head>
<body>
    <header>
        <div class="center">
            <div class="logo">
                <a href="index.html">На главную</a>
href="https://drive.google.com/drive/folders/1ZugqcHTbfqaNu64ULXxcGxFxfEIRDbx
z">GoogleDrive</a> -->
            </div>
            <div class="menu">
                <a href="contents.html">Содержание</a>
                <a href="introduction.html">Введение</a>
                <!-- <a href="chapter1.html">Глава I</a> -->
                <a href="chapter2.html">Глава II</a>
                <a href="conclusion.html">Заключение</a>
                <a href="appendix.html">Приложения</a>
                <a href="literature.html">Список литературы</a>
```

```
<!-- <a
href="https://github.com/BoginskiyVV/webpageDiploma">Репозиторий Github</a>
               <a class="btn-menu" href="#">Ссылка</a> -->
           </div>
       </div>
       <!--Center-->
   </header>
   <section class="main">
       <div class="center">
           <div class="main cta">
               <h2>ГЛАВА I</h2>
                   <b>Глава І. Теоретическая
часть.</b><br><br></b></br></r>
                  <br>
                       <b>2.1. Язык Python для
поиска информации в сети Интернет и парсинга
                      сайтов.</b><br>
                       <b>2.1.1. Общая информация
по языку Python и парсингу сайтов.</b>
                     В современном информационном обществе
объем данных, представленных в сети
                  Интернет, увеличивается с каждым днем.
                  Использование этих данных может стать ценным ресурсом для
решения множества задач, включая
                  извлечение и дальнейшее
                  использование информации, анализ данных, машинное
обучение и другие. Большинство данных Интернете,
                  доступны
                  в виде веб-страниц, и для их эффективного использования
необходимо уметь собирать и обрабатывать
                  нужную
                  информацию. <br>
                     Парсинг веб-страниц - это процесс
извлечения информации из такой страницы, анализа
                  ее содержимого
                  и преобразования в удобный формат для дальнейшей
обработки, хранения и более глубокого анализа.
                  Python - это
                  мощный, современный, гибкий, высокоуровневый,
интерпретируемый язык программирования, который
                  примечателен
```

```
своей простотой, читаемостью и обширной экосистемой
библиотек и инструментов. Python стал одним из
                  наиболее
                  популярных языков программирования для разработки веб-
приложений, анализа данных, искусственного
                  интеллекта
                  и других областей. <br>
                     Язык Python является отличным выбором
для работы с данными в Интернете. Благодаря
                  множеству библиотек
                  и инструментов, доступных на этом языке, поиск информации
и парсинг веб-страниц доступен для
                  широкого круга
                  как продвинутых, так и начинающих специалистов.
Независимо от того, стоит ли просто вопрос получения
                  информации для анализа данных, обновления содержимого
сайта или выполнения других задач, Python
                  готов
                  предоставить все необходимые инструменты для успешной
реализации поисково-аналитического комплекса
                  или реализации иного проекта. <br>
                     Основные инструменты для парсинга веб-
страниц на языке Python:<br>
                     - Библиотека requests: используется для
отправки НТТР-запросов и получения
                  содержимого страницы. <br>
                      - Библиотека BeautifulSoup:
предоставляет удобные методы для разбора
                  HTML-кода и извлечения данных. <br>
                      - Библиотека lxml: используется
для парсинга XML-документов и работы с
                  XPath-запросами. <br>
                     - Библиотека Scrapy: предоставляет
инструменты для создания веб-скрейперов и
                  парсеров с возможностью
                  обхода и сбора информации с нескольких страниц. <br>
                     - Библиотека Selenium: мощный и
популярный инструмент не только для парсинга
                  сайтов, но и для решения
                  задач в области разработки и тестирования веб-
страниц. <br>
                  <br>>
                      <b>2.1.2. Основы использования
библиотеки requests.</b><br>
```

```
    Библиотека requests является
одной из самых популярных библиотек для
                   отправки НТТР-запросов
                   и получения данных из сети в языке программирования
Python. Она предоставляет простой и удобный
                   интерфейс для работы
                   с веб-ресурсами, что делает ее идеальным выбором для
парсинга сайтов. С ее помощью возможно
                   эффективно и гибко собирать
                   нужную информацию с веб-страниц и использовать ее для
различных задач, таких как анализ данных,
                   мониторинг или
                   автоматизация процессов. <br>
                      Основным способом получения веб-страниц
является отправка GET-запросов. Библиотека
                   requests
                   предоставляет функцию get, которая позволяет отправлять
GET-запросы на веб-ресурсы и получать
                   содержимое
                   страницы. Возвращаемый объект Response содержит различные
данные о запросе и ответе сервера, включая
                   статус-код,
                   заголовки и тело ответа. <br>
                   Изображение № 1. Пример отправки GET-запроса и вывода
содержимого страницы<br>
                   <img src="src/chapter1-1.png"><br>
                     В примере, представленном на изображении №1
мы отправляем GET-запрос на
                   страницу https://example.com и выводим статус-код ответа
и содержимое страницы.<br>
                       Также часто для получения
нужной информации с веб-страницы требуется
                   передать параметры в запросе (например, для поиска).
                   Библиотека requests позволяет передавать параметры
запроса в виде словаря с помощью аргумента params
                   функции get.<br>
                   <br>
                   Изображение № 2. Пример отправки GET-запроса с аргументом
params<br>
                   <img src="src/chapter1-2.png"><br>
                   <br>>
```

```
   В примере, приведенном на
изображении №2 мы передаем параметры запроса в
                  словаре
                  params и отправляем GET-запрос на
https://example.com/search. Затем мы выводим URL-адрес запроса,
                  чтобы
                  убедиться, что параметры были переданы правильно. <br>
                     Иногда веб-серверы требуют определенных
заголовков в запросе для качественной
                  обработки запроса. Библиотека requests позволяет
добавлять заголовки в запрос с помощью аргумента
                  headers
                  функции get.<br>
                  <br>>
                  Изображение № 3. Пример отправки GET-запроса с
использованием заголовка<br>
                  <img src="src/chapter1-3.png"><br>
                  <br>
                     В примере, приведенном на
изображении №3 мы добавляем к нашему GET-запросу
                  заголовок
                  User-Agent, который говорит серверу о том, что наш запрос
идет от пользователя с браузером
                  Mozilla/5.0.<br>
                     При парсинге веб-страницы важно
предусмотреть возможные ошибки, такие как
                  недоступность
                  сервера или неправильные параметры запроса. Библиотека
requests предоставляет возможность обработки
                  ошибок
                  с помощью исключений. <br>
                  <hr>>
                  Изображение № 4. Пример обработки ошибок<br>
                  <img src="src/chapter1-4.png"><br>
                  <br>
                     В примере, приведенном на
изображении №4, мы используем блоки try-except для
                  обработки
                  возможных ошибок. Если возникает ошибка НТТР, соединения
или запроса, мы выводим соответствующее
                  сообщение об
                  ошибке.<br>
                     Виблиотека requests предоставляет
множество других функций и возможностей
```

```
для более
                  гибкого и продвинутого парсинга веб-страниц, которые
будут представлены в практической части работы
                  и приложениях. <br>
                      <b>2.1.2. Основы использования
библиотеки BeautifulSoup</b><br>
                     Библиотека BeautifulSoup предоставляет
удобные инструменты для парсинга HTML и XML
                  кода сайтов, облегчая извлечение информации и навигацию
по структуре веб-страниц. Использование
                  библиотеки
                  BeautifulSoup значительно упрощает процесс парсинга веб-
страниц на языке Python. Благодаря удобным
                  методам
                  и инструментам, предоставляемым этой библиотекой, имеется
возможность быстро и эффективно извлечь
                  нужную
                  информацию из HTML кода страницы. <br>
                      Прежде чем начать использовать
BeautifulSoup, необходимо установить ее.
                  Установка
                  библиотеки производится с помощью следующей команды
pip:<br>
                     <span style="color:</pre>
#ff6600;">"pip install beautifulsoup4"</span><br>
                     После установки мы импортируем
библиотеку BeautifulSoup в свой код: <br>
                     <span style="color:</pre>
#ff6600;">"from bs4 import BeautifulSoup"</span><br>
                    Для парсинга веб-страницы сначала нужно
получить HTML код этой страницы. Для этого
                  необходимо использовать описанную ранее библиотеку
requests или любой другой способ, который
                  позволяет
                  получить HTML код страницы (Изображение №5).<br>
                  <br>
                  Изображение № 5<br>
                  <img src="src/chapter1-5.png"><br>
                     После получения HTML кода
страницы, мы можем создать объект BeautifulSoup
```

для

```
парсинга этого кода. В качестве второго аргумента в
конструкторе BeautifulSoup указывается парсер,
                  которым будет обрабатываться HTML код. Наиболее часто
используемые парсеры - это 'html.parser',
                  'lxml' и 'html5lib' (Изображение №6).<br>
                  Изображение № 6<br>
                  <img src="src/chapter1-6.png"><br>
                      Библиотека BeautifulSoup
предоставляет множество методов для извлечения
                  информации
                  из HTML кода. Некоторые из основных методов: <br/>
                      - find(): Ищет первый элемент,
удовлетворяющий заданным критериям. <br>
                      - find_all(): Ищет все элементы,
удовлетворяющие заданным критериям. <br>
                      anbsp; - get_text(): Получает текстовое
содержимое элемента. <br>
                      - get(): Получает значение
атрибута элемента.<br>
                      - select(): Выполняет поиск
элементов с помощью CSS-селекторов. <br>
                     В примере на изображении №7 мы
ищем заголовок страницы с помощью метода find
                  и выводим его текстовое содержимое. Затем мы итерируемся
по всем ссылкам на странице, используя
                  метод find all, и выводим URL-адрес каждой ссылки. <br>
                  <br>
                  Изображение № 7<br>
                  <img src="src/chapter1-7.png"><br>
                      Также библиотека BeautifulSoup
позволяет осуществлять навигацию по структуре
                  HTML кода. Так мы можем перемещаться по родительским и
дочерним элементам и выполнять поиск
                  элементов
                  на основе их свойств и отношений. <br>
                  <br>>
                  Изображение № 8<br>
                  <img src="src/chapter1-8.png"><br>
                  <br>
                     В примере на изображении №8 мы
сначала находим элемент "body" и выводим его
```

```
дочерние
                  элементы. Затем мы ищем все элементы "div" с классом
"container" и выводим их. Наконец, мы находим
                  следующий
                  соседний элемент после "header" и выводим его. <br>
                      Библиотека BeautifulSoup
предоставляет еще множество других методов и
                  возможностей
                  для более гибкого и продвинутого парсинга HTML кода,
которые будут представлены в практической части
                  работы
                  и приложениях. <br>
                  <br>
                      <b>2.1.3. Основы использования
библиотеки Scrapy</b><br>
                    Scrapy - это мощный и гибкий фреймворк для
создания веб-скрейперов и парсеров на
                  языке программирования Python. Он предоставляет множество
инструментов и функций для автоматизации
                  процесса
                  сбора данных с веб-страниц и обхода нескольких страниц
одновременно. Рассмотрим основы использования
                  библиотеки Scrapy для парсинга сайтов. <br>
                    Прежде чем начать использовать Scrapy,
необходимо установить ее. Установка библиотеки
                  производится с помощью следующей команды pip: <br>
                     <span style="color:</pre>
#ff6600;">"pip install scrapy"</span><br>
                     После установки мы импортируем
библиотеку Scrapy в свой код:<br>
                     <span style="color:</pre>
#ff6600;">"import scrapy"</span><br>
                     Для создания парсера с помощью
Scrapy необходимо создать новый проект Scrapy
                  с помощью командной строки:<br>
                     <span style="color:</pre>
#ff6600;">"scrapy startproject myproject"</span><br>
                    После создания проекта, необходимо создать
спайдера - класс, который определяет
                  логику парсинга веб-страниц. В методе parse спайдера вы
можете определить, какие данные нужно
                  собрать и какие
                  действия нужно совершить на странице. <br>
```


>

```
Изображение № 9.<br>
                   <img src="src/chapter1-9.png"><br>
                   <br>
                       На примере, приведенном на
изображении №9 мы создаем новый класс MySpider,
                   устанавливаем его имя и стартовый URL-адрес. В методе
parse мы извлекаем информацию из текущей
                   страницы,
                   используя CSS-селекторы. Затем мы вызываем метод follow
объекта ответа, чтобы перейти на другую
                   страницу
                   и продолжить обработку с помощью метода
parse_other_page.<br>>
                      После создания спайдера
необходимо запустить парсер с помощью команды scrapy
                   crawl:<br>
                      <span style="color:</pre>
#ff6600;">"scrapy crawl example spider"</span><br>
                      Scrapy автоматически выполняет запросы,
следует по ссылкам и вызывает методы
                   парсера
                   для каждой страницы. Можно настроить Scrapy для обхода
нескольких страниц, установки задержек или
                   отправки
                   пользовательских заголовков запроса. <br>
                      Спайдеры Scrapy предоставляют
возможности для сохранения собранных данных.
                   Для этого можно использовать различные результирующие
пайплайны, чтобы обрабатывать, фильтровать и
                   сохранять
                   данные в различных форматах, таких как CSV, JSON или
базах данных. <br>
                   <br>
                   Изображение № 10.<br>
                   <img src="src/chapter1-10.png"><br>
                      В примере на изображении №10 мы
инициализируем экспортер CSV и открываем
                   файл data.csv,
                   в который будут сохраняться данные. Затем мы вызываем
метод export item для сохранения элементов на
                   каждой
                   странице. По завершению работы мы вызываем метод
finish exporting.<br>
```

```
<br>
                      <b>2.1.4. Основы использования
библиотеки Selenium</b><br>
                    Библиотека Selenium - это мощный инструмент
для автоматизации веб-браузера,
                  который позволяет управлять браузером и взаимодействовать
с веб-страницами через код на языке
                  программирования
                  Python. Она является одним из наиболее популярных
инструментов для парсинга сайтов, особенно тех, в
                  которых
                  осуществляется взаимодействие с JavaScript. <br>
                      Благодаря ее возможностям
управление и взаимодействовие с веб-страницами
                  представляется
                  высокоэффективным, особенно в части сбора необходимой
информации и автоматизации рутинных задач,
                  связанных
                  с парсингом и получением данных. В сочетании с другими
инструментами Python, библиотека Selenium
                  предоставляет
                  широкие возможности для парсинга сайтов и работы с
данными из сети Интернет. <br>
                     Перед началом использования
библиотеки Selenium, необходимо установить
                  драйвер браузера,
                  который будет использоваться для автоматизации. Драйверы
могут быть установлены и настроены согласно
                  выбору
                  браузера. Например, для браузера Chrome необходимо
установить ChromeDriver, а для Firefox -
                  GeckoDriver.
                  После установки драйвера нужно убедиться, что путь к нему
добавлен в переменную среды PATH. <br>
                      Кроме того, также понадобится
установить библиотеку Selenium через команду
                  pip:<br>
                     <span style="color:</pre>
#ff6600;">"pip install selenium"</span><br>
                     Далее, прежде всего, импортируем
необходимые модули Selenium в код
                  Python: <br>
                     <span style="color:</pre>
#ff6600;">"from selenium import webdriver"</span><br>
```

```
    Затем создается экземпляр класса
веб-драйвера для выбранного браузера: <br>
                     <span style="color:</pre>
#ff6600;">"driver = webdriver.Chrome()
                      # Создание веб-драйвера Chrome"</span><br>
                      У объекта веб-драйвера есть
множество методов для управления веб-страницами.
                  Например, метод get используется для перехода на
определенную веб-страницу:<br>
                      <span style="color:</pre>
#ff6600;">"driver.get("https://example.com")
                      # Переход на страницу example.com"</span><br>
                     Meтоды класса webdriver также
позволяют выполнять различные действия на
                  веб-страницах,
                  такие как заполнение полей формы, нажатие кнопок,
скроллинг страницы и многое другое. <br>
                      Основной метод получения
информации из веб-страницы с использованием
                  Selenium -
                  это метод find element by, который находит элемент на
странице на основе различных атрибутов (Class
                  Name,
                  CSS Selector, ID, Link Text и другие). Он используется
для нахождения нужной информации на
                  веб-странице. <br>
                     <span style="color:</pre>
#ff6600;">"element =
                      driver.find element by css selector('#my-element')
                      # Найти элемент по селектору CSS "</span><br>
                     <span style="color:</pre>
#ff6600;">"print(element.text)
                      # Извлечь текст элемента"</span><br>
                   В представленном выше примере мы находим элемент с
помощью метода
                  find element by css selector, передавая ему CSS-селектор
элемента. Затем мы выводим текстовое
                  содержимое найденного элемента. <br>
                     После завершения работы
необходимо закрыть окно браузера, чтобы освободить
                  ресурсы.
                  Такое действие реализуется с помощью метода close: <br/>
                     <span style="color:</pre>
#ff6600;">"driver.close() # Закрытие окна
```

```
браузера"</span><br>
                     Также возможно закрыть полностью
веб-драйвер с помощью метода quit:<br>
                     <span style="color:</pre>
#ff6600;">"driver.quit() # Полное завершение работы
                     веб-драйвера"</span><br>
                    Важно помнить, что закрывать окно
браузера нужно после завершения всех
                 нужных
                 операций с веб-страницей. <br>
                     По умолчанию браузер Selenium
работает в видимом режиме, и можно видеть,
                 как он выполняет действия на веб-странице. Однако
библиотека Selenium также предоставляет
                 возможность
                 работы в фоновом режиме с помощью драйвера PhantomJS или
Headless Chrome. <br>
                    Пример использования
PhantomJS:<br>
                     <span style="color:</pre>
#ff6600;">"from selenium.webdriver import
                     PhantomJS"</span><br>
                    <span style="color:</pre>
#ff6600;">"driver = PhantomJS() # Создание экземпляра
                     PhantomJS
                     веб-драйвера"</span><br>
                    <span style="color:</pre>
#ff6600;">"driver.get("https://example.com")
                     # Переход на страницу example.com"</span><br>
                 <hr>>
                     <b>2.2. Средства хранения данных
при парсинге сайтов</b><br>
                     <b>2.2.1. Использование CSV при
парсинге сайтов</b><br>
                    Одним из способов организации и
сохранения данных, полученных полученных при
                 парсинге web-страниц, является использование SCV-формата
(Structured Comma Separated Values). <br>
                     SCV-формат представляет собой
специальный тип текстового файла, который
                 используется
                 для структурированного хранения данных в виде таблицы.
Файл SCV содержит данные, разделенные
                 запятыми
```

```
(или другим разделителем), что позволяет легко читать и
обрабатывать информацию. <br>
                     Данные HTML-кода страницы,
извлеченные с помощью, например, такой библиотеки
                  Python, как BeautifulSoup, возможно сохранять в файл SCV
с использованием одноименной библиотеки
                  CSV.
                  Рассмотрим основные преимущества и недостатки данного
подхода. <br>
                     Преимущества использования SCV в
парсинге сайтов:<br>
                      • Простота в использовании.
Формат SCV предоставляет простую и понятную
                  структуру
                  данных, которая легко читается и записывается с помощью
Python. Это делает процесс парсинга и
                  сохранения
                  данных максимально удобным и эффективным. <br>
                     • Универсальность. Файлы SCV могут быть
открыты различными программами,
                  включая электронные таблицы, что облегчает работу с
данными. Благодаря доступности SCV-формата,
                  полученные данные могут быть использованы в других
приложениях и инструментах для анализа,
                  визуализации
                  и обработки информации.<br>
                      « Сохранение структуры данных.
Файлы SCV могут сохранять не только значения,
                  но и структуру данных, что позволяет сохранять
соответствие между различными столбцами и строками.
                  Это значительно упрощает дальнейшую обработку данных и
предоставляет возможность легко извлекать
                  необходимые значения. <br>
                      возможность автоматизации. С
использованием Python и библиотеки для работы
                  с SCV можно создать скрипт, который будет автоматически
парсить веб-страницы и сохранять данные в
                  SCV-файлы.
                  Это позволяет значительно сэкономить время и упростить
работу с большим объемом данных. <br>
                  <br>>
                        Недостатки использования SCV при
```

парсинге сайтов:


```
    • Ограниченность структуры
данных. SCV-формат предоставляет ограниченные
                   возможности для структурирования данных. Например,
сложные связи между таблицами или специфические
                   типы данных могут быть сложно или невозможно представить
в формате SCV.<br>
                      • Ограничения на объем данных. SCV-
формат является текстовым и не предназначен
                   для хранения больших объемов данных. В случае работы с
очень большими массивами информации,
                   использование
                   SCV может снизить производительность и эффективность
обработки данных.<br>
                      • Возможные проблемы с кодировкой. В
некоторых случаях могут возникнуть проблемы
                   с кодировкой данных при сохранении и чтении SCV-файлов.
Это может привести к искажению данных или
                   их неправильному представлению. <br>
                       Отметим, что использование SCV-
формата при парсинге сайтов с помощью Python
                   предоставляет достаточно удобный и эффективный способ
организации и сохранения полученных данных.
                   Однако, необходимо учитывать некоторые ограничения
данного формата, как на структуру данных,
                   так и на возможное возникновение проблем с
кодировкой. <br>
                   <br>
                       <b>2.2.2. Использование JSON при
парсинге сайтов</b><br>
                      JSON (JavaScript Object Notation)
- это формат обмена данными, основанный на
                   языке JavaScript. Он широко используется веб-
разработчиками для передачи и обмена структурированными
                   данными между клиентской и серверной сторонами
приложения. При парсинге сайтов, JSON используется
                   извлечения и преобразования данных, полученных из HTML-
разметки в более удобный и легкодоступный
                   формат. <br>
                       JSON предоставляет удобное и
легкочитаемое представление данных,
                   которое можно использовать для анализа и дальнейшей
обработки. Он позволяет сократить объем кода и
```

сделать

```
процесс парсинга более эффективным и гибким. <br>
                      Для начала, необходимо получить
HTML-код страницы. Это можно сделать,
                  используя библиотеку HTTP-запросов, такую как requests,
которая была упомянута выше. После получения
                  HTML-кода, можно использовать BeautifulSoup, 1xml или
друние парсеры, чтобы извлечь данные из
                  HTML-разметки. <br>
                      Затем, если данные, которые нужно
извлечь, представлены в формате JSON
                  на странице, мы можем использовать методы работы с JSON
для их извлечения. JSON-данные обычно
                  представляются в виде пар "ключ-значение" и могут быть
представлены в виде объектов, массивов,
                  чисел, строк и логических значений. <br>
                     В простых случаях, когда данные
находятся в корневом уровне страницы,
                  можно просто обратиться к ним через общий для большинства
языков программирования синтаксис
                  обращения к элементам словаря. <br>
                      Следует учитывать, что в
некоторых случаях данные могут находиться
                  во вложенных структурах и требуют достаточно сложных
запросов. В таких случаях следует применить
                  такие методы работы с JSON, как использование пути к
элементу или циклов для обхода всех элементов
                  массива или объекта.<br>
                     Кроме того, необходимо учитывать,
что при парсинге сайтов,
                  некоторые веб-страницы могут быть динамически построены с
помощью JavaScript, что может усложнить
                  процесс извлечения данных из JSON. В таких случаях может
потребоваться использование
                  инструментов для автоматического выполнения JavaScript,
таких как Selenium WebDriver. <br>
                  <hr>>
                      <b>2.2.3. Использование SQL при
парсинге сайтов</b><br>
                     Для эффективного хранения и
организации данных, собранных в процессе
                  парсинга веб-страниц часто используется язык
структурированных запросов SQL (Structured
```

Query Language) и реляционные базы данных.

Перед началом работы с SQL для хранения данных, нужно выбрать

соответствующую базу данных, которая будет отвечать требованиям проекта. Существуют различные

реляционные базы данных, такие как MySQL, PostgreSQL, SQLite и другие, которые предоставляют

различные функциональности и возможности. Выбор конкретно системой управления базой данных будет

зависеть от предпочтений команды разработчиков,

требований к масштабируемости, производительности

и некоторых других факторов.

Oдним из основных преимуществ SQL является его способность обеспечивать

структурированное хранение данных. SQL базы данных состоят из таблиц с заданными столбцами и типами

данных для каждого столбца. Это позволяет легко

организовать данные и обеспечить соответствие

структуре

данных, что облегчает работу с ними.

В то время как CSV и JSON

являются неструктурированными форматами данных,

которые сохраняют информацию в виде текста без явного определения структуры данных. В CSV данные

хранятся в простом текстовом формате с разделителями, а в JSON данные хранятся в виде пар

"ключ-значение".

Это может усложнить управление и обработку данных при использовании этих форматов.

SQL предоставляет средства для выполнения мощных и гибких запросов к данным.

Имеется возможность использовать язык SQL для извлечения нужных данных, применять фильтры,

сортировать

результаты и связывать данные из разных таблиц. Благодаря этому, SQL дает возможность быстро

получить

нужные данные и осуществить сложные операции с ними.
 С другой стороны, в CSV и JSON

форматах отсутствуют встроенные механизмы

запросов

и фильтрации данных. Для извлечения отдельных данных или выполнения сложных операций придется обрабатывать

```
данные вручную с использованием других инструментов или
кода Python. В отдельных случаях может быть
                   более
                   трудоемким и менее эффективным в сравнении с
использованием SQL запросов. <br>
                       SQL базы данных обеспечивают
поддержку целостности данных и механизмы
                   контроля
                   доступа. Имеется возможность определить ограничения на
значения в таблицах для обеспечения
                   целостности
                   данных, а также настроить права доступа для пользователей
и ролей. Это значительно упрощает
                   управление
                   и контроль данных, особенно в случаях, когда нужно
обрабатывать чувствительные данные или
                   предоставлять
                   доступ к базе данных другим пользователям. <br>
                       С другой стороны, в CSV и JSON
форматах данных нет встроенной поддержки
                   целостности данных или контроля доступа, что может
привести к выполнению таких операций вручную,
                   оказаться трудоемким и потенциально приводить к
ошибкам. <br>
                       SQL базы данных предоставляют
хорошую масштабируемость и производительность
                   для хранения и обработки больших объемов данных. Они
обладают эффективными механизмами работы с
                   данными,
                   оптимизацией запросов и поддержкой индексов для ускорения
выполнения запросов и улучшения
                   производительности.
                      В то время как CSV и JSON форматы
имеют свои преимущества, они не всегда
                   подходят
                   для хранения и обработки больших объемов данных. Они
могут стать неэффективными при работе
                   с большими файлами, а также создавать сложности при
выполнении операций с данными. <br>
                       <b>2.2.4. Использование Наdoop
при парсинге сайтов</b><br>
```

Hadoop - это фреймворк для

обработки больших объемов данных на

```
распределенных
```

кластерах. В его основе лежит концепция MapReduce, которая позволяет распараллеливать вычисления

и значительно ускорять обработку данных.

Парсинг сайтов на Python с

использованием Hadoop может быть полезен

при работе с большим количеством данных, которые требуют распределенной обработки. Например,

если необходимо обработать данные с нескольких тысяч или даже миллионов веб-страниц, обработать

петабайты данных, Hadoop может помочь распараллелить процесс получения и обработки данных,

что может ускорить работу нашего поисково-аналитического программного комплекса.

Для совместного использования
Python и Hadoop можно воспользоваться

такими библиотеками, как Hadoop Streaming и Pydoop.
 Приведем ряд преимуществ Hadoop,

которые могут быть учтены при использовании

инструмента для парсинга сайтов:

• Масштабируемость: Наdоор позволяет

обрабатывать большие объемы данных путем

распределения вычислений на большое количество узлов кластера. Это позволяет обрабатывать

и анализировать данные веб-страниц эффективно и масштабировать решение по мере увеличения

количества обрабатываемых сайтов.

* Параллелизм: Наdoop использует концепцию MapReduce, которая позволяет

распараллеливать процесс обработки данных. Данные вебстраницы могут быть разделены на части

и обработаны параллельно на разных узлах кластера, что ускоряет общее время обработки.

• Отказоустойчивость: Hadoop обеспечивает отказоустойчивость за счет

репликации данных на разных узлах кластера. Это означает, что если один узел выходит из строя,

данные сохраняются на других узлах и обработка может продолжаться без потери данных.

2.3. Использование языка Python для анализа данных

2.3.1. Общая информация по возможностям языка Python

для анализа данных

Python является одним из наиболее популярных и широко используемых языков

программирования в мире. Он обладает простым и интуитивно понятным синтаксисом, что делает его

идеальным выбором для анализа данных. Python является мощным и гибким языком программирования,

который предоставляет широкий спектр инструментов и библиотек для анализа данных.

Он легко изучается и его использование для анализа данных может значительно упростить

и ускорить процесс.

Baжно отметить, что Python

является языком с открытым исходным кодом,

что означает, что он бесплатен и доступен для всех. Это делает его использование актуальным

для широкого круга пользователей и организаций, и позволяет использовать его для различных

целей, включая научные исследования, анализ данных, разработку приложений и многое другое.

Oдной из самых популярных библиотек Python для работы с данными

является Pandas. Данная библиотека предоставляет удобный и гибкий интерфейс для работы

с таблицами данных, анализа и манипулирования ими. Pandas предлагает функциональность,

которая позволяет легко выполнять операции, такие как фильтрация, сортировка, объединение

и агрегирование данных.

Eще одной мощной библиотекой Python для анализа данных является

NumPy. Данная библиотека предоставляет поддержку для многомерных массивов, а также

широкий набор функций для работы с ними. NumPy позволяет выполнять математические

операции, такие как сложение, умножение и

тригонометрические функции, на массивах

данных. Это позволяет рассматривать NumPy как идеальный инструмент для работы

с большими объемами числовых данных.

Для визуализации данных Python

предлагает библиотеку

Matplotlib. Данная библиотека позволяет создавать различные типы графиков,

включая линейные, столбчатые, круговые диаграммы, диаграммы рассеяния и

другие. Matplotlib также предоставляет возможность настройки внешнего вида

графиков, включая метки осей, легенды и цвета. В сочетании с Pandas и NumPy,

Matplotlib делает Python мощным инструментом для визуализации и анализа данных.

Также стоит отметить библиотеку Seaborn, построенную на Matplotlib и

предназначенную для создания статистических визуализаций.

Seaborn хорошо

интегрирован с Pandas и позволяет делать выразительные

построения наборов

статистических данных, понятные как исследователю, так и конечному пользователю

аналитического продукта.

Eще одной популярной библиотекой

для анализа данных

на Python является SciPy. SciPy предоставляет набор

функций для решения

сложных математических и научных задач. Он включает в

себя функции для

оптимизации, статистики, интерполяции, алгебры и многое

другое. SciPy

значительно расширяет возможности Python для анализа

данных и делает

его подходящим для широкого спектра приложений.

Для машинного обучения и искусственного интеллекта Python предлагает несколько

популярных библиотек, таких как TensorFlow, Keras и Scikit-Learn. TensorFlow

является мощной библиотекой для создания и обучения нейронных сетей. Keras

предоставляет простой и интуитивно понятный интерфейс для создания моделей

машинного обучения. Scikit-Learn является библиотекой для машинного обучения,

предоставляющей реализацию широкого спектра алгоритмов, таких как классификация,

регрессия, кластеризация и многое другое.

Python также имеет широкую поддержку для работы с базами

данных. Он предоставляет библиотеки, такие как SQLAlchemy, для работы с различными

типами баз данных, включая SQL и NoSQL. Python также имеет возможности для

работы с API и получения данных из различных источников сети Интернет и

различных веб-сервисов, что частично было описано в предыдущих разделах.

Также важно отметить, что одним из преимуществ

использования Python для анализа данных является огромное сообщество

разработчиков и наличие качественно составленной технической документации

для большинства инструментов языка. При возникновении вопросов у разработчиков

и аналитиков всегда есть возможность обратиться к сообществу и найти ответы.

сиблиотеки Pandas для

анализа данных

Pandas предоставляет удобный и мощный инструментарий

для работы с табличными данными и выполнения различных операций над ними.

В этой статье мы рассмотрим основные возможности и функции библиотеки Pandas,

которые делают ее незаменимым инструментом для анализа данных.

Ocновным объектом в библиотеке
Pandas является DataFrame

- двумерный массив данных, представляющий собой таблицу с метками строк и столбцов.

DataFrame позволяет хранить и манипулировать данными различных типов, включая числа,

строки, даты и другие.

Oдной из первых задач при работе с данными является их загрузка.

Pandas предоставляет множество функций для чтения данных из различных источников, включая

```
CSV-файлы, Excel, SQL-базы данных и даже веб-страницы.
Например, функция read csv()
                  позволяет загрузить данные из CSV-файла и создать
DataFrame: <br>
                  <br>
                  Изображение № 11<br>
                  <img src="src/chapter1-11.png"><br>
                   <br>
                     После загрузки данных вы
становится возможным выполнение различных
                  операций над ними. Например, можно узнать размерность
данных с помощью атрибута shape:<br>
                  <br>
                  Изображение № 12<br>
                   <img src="src/chapter1-12.png"><br>
                     Получить первые или последние
строки данных можно с помощью методов
                  head() и tail():<br>
                  <hr>>
                  Изображение № 13<br>
                  <img src="src/chapter1-13.png"><br>
                   <br>
                      Pandas предоставляет множество
функций для фильтрации, сортировки
                  и манипуляции данными. Например, можно отфильтровать
данные, выбрав только те строки,
                  которые удовлетворяют определенным условиям: <br>
                  <br>
                  Изображение № 14<br>
                  <img src="src/chapter1-14.png"><br>
                      Можно выполнять агрегирование
данных, например, подсчитывать
                  среднее значение, максимальное или минимальное значение
столбца с помощью метода describe():<br>
                  <br>>
                  Изображение № 15<br>
                  <img src="src/chapter1-15.png"><br>
                     Pandas также предлагает удобные
функции для группировки данных, выполнения
                  операций при соединении таблиц и многое другое. Например,
функция groupby() позволяет сгруппировать
```

```
операцию над каждой группой:<br>
                   <br>
                   Изображение № 16<br>
                   <img src="src/chapter1-16.png"><br>
                      Важной особенностью Pandas
является способность обрабатывать отсутствующие
                   значения, которые могут быть присутствовать в данных.
Pandas предоставляет функции для замены
                   отсутствующих значений, удаления строк или столбцов с
отсутствующими значениями и другие функции
                   для работы с пропущенными данными. <br>
                       Pandas также интегрирован с
другими популярными библиотеками Python,
                   такими как NumPy и Matplotlib, что позволяет более
эффективно выполнять анализ данных и
                   визуализацию результатов. Можно использовать функции
NumPy для выполнения вычислений над
                   данными Pandas, a Matplotlib для создания графиков и
визуализации результатов анализа. Более
                   глубокие представления о Pandas будут реализованы в
практической части работы и приложениях. <br>
                       <b>2.3.3. Основы использования
библиотек Matplotlib и
                       Seaborn для анализа данных</b><br>
                   <br>
                      Matplotlib и Seaborn - это две
мощные библиотеки для визуализации данных
                   на языке программирования Python. Они предлагают широкие
возможности для создания графиков
                   и визуализации информации с минимальными усилиями.
Использование этих библиотек в анализе
                   данных помогает представить результаты аналитической
работы более наглядно и понятно. <br>
                      Данные библиотек предоставляют
широкие возможности выбора наиболее
                   подходящего графического инструмента, который лучше всего
подходит для конкретных потребностей
                   и задач. Использование этих библиотек вместе с другими
инструментами для анализа данных, такими
                   как библиотека Pandas, помогает лучше понять и
интерпретировать данные. <br>
```

данные по определенному столбцу и выполнить агрегирующую

```
    Matplotlib предоставляет широкий
спектр возможностей для создания
                  различных типов графиков, включая линейные, столбчатые,
круговые диаграммы, диаграммы
                  рассеяния и т.д., а также предлагает гибкую настройку
внешнего вида графиков, включая настройку
                  легенд, меток осей, цветов и т.д.<br>
                      Для создания графиков с помощью
Matplotlib, первым шагом является
                  импорт необходимых модулей:<br>
                  <br>
                  Изображение № 17<br>
                  <img src="src/chapter1-17.png"><br>
                  <hr>>
                      Затем вы можете создать простой
график, используя функции plot()
                  и show():<br>
                  <br>
                  Изображение № 18<br>
                  <img src="src/chapter1-18.png"><br>
                     Matplotlib также предлагает
множество возможностей для настройки
                  внешнего вида графиков. Существует возможность добавить
метки для осей, заголовки,
                  легенды и многое другое с помощью соответствующих
функций:<br>
                  <hr>>
                  Изображение № 19<br>
                  <img src="src/chapter1-19.png"><br>
                      Seaborn, основанная на
Matplotlib, позволяет легко создавать
                  привлекательные графики с помощью простых кодовых
конструкций. Для использования
                  Seaborn сначала необходимо импортировать необходимые
модули:<br>
                  <hr>>
                  Изображение № 20<br>
                  <img src="src/chapter1-20.png"><br>
                  <br>>
                      Затем можно использовать функции
Seaborn для создания различных
```

```
типов графиков, таких как столбчатые диаграммы,
гистограммы, ящики с усами и др. Например,
                  создать столбчатую диаграмму с помощью функции
barplot():<br>
                  <br>
                  Изображение № 21<br>
                  <img src="src/chapter1-21.png"><br>
                  <br>
                      Seaborn также предлагает функции
для визуализации статистической
                  информации, например графики ящика с усами для
отображения распределения данных и
                  показателей центра и размаха:<br>
                  <hr>>
                  Изображение № 22<br>
                  <img src="src/chapter1-22.png"><br>
                     Волее глубокие представления о
библиотеке Matplotlib будут
                  реализованы в практической части работы и
приложениях.<br>
                  <hr>>
                      <b>2.4. Использование SQL для
анализа данных</b><br>
                      SQL является одним из наиболее
распространенных языков
                  программирования для работы с базами данных. Он позволяет
эффективно извлекать д
                  анные из базы данных, выполнять сложные запросы,
агрегирующие функции и другие операции.
                      SQL предоставляет возможность
извлечения, обновления, добавления
                  и удаления данных из базы данных. Он основан на наборе
команд и операторов, которые позволяют
                  работать с различными типами данных, такими как строки,
числа, даты и другие. SQL поддерживает
                  операции выборки данных (SELECT), вставки данных
(INSERT), изменения данных (UPDATE) и удаления
                  данных (DELETE).
                      Оператор SELECT является основным
для анализа данных в SQL. Он позволяет
                  выбрать данные из одной или нескольких таблиц базы
```

данных, исходя из определенных условий.

```
Ключевое слово SELECT заставляет SQL вернуть определенный
набор данных, определенный в запросе.
                  Вот простой пример оператора SELECT: <br>
                  <br>
                  Изображение № 23<br>
                  <img src="src/chapter1-23.png"><br>
                  <br>
                     В примере, приведенном на
изображении №22 мы выбираем все данные
                  из таблицы employees. Звездочка (*) указывает, что мы
выбираем все столбцы из таблицы. Можно
                  также указать конкретные столбцы, которые требуется
выбрать в запросе.
                      Оператор WHERE позволяет
фильтровать данные в запросе на основе
                  определенных условий. Например: <br>
                  <br>
                  Изображение № 24<br>
                  <img src="src/chapter1-24.png"><br>
                     В запросе, приведенном на
изображении №23, мы выбираем только те
                  строки таблицы employees, в которых значение столбца age
больше 30. Мы можем комбинировать
                  различные условия, используя логические операторы, такие
как AND, OR, NOT.<br>
                      SQL также предлагает возможности
агрегации данных, такие
                  как сумма, среднее и максимальное значение столбца, или
подсчет количества строк,
                  удовлетворяющих определенным условиям. Например: <br/>
                  <hr>>
                  Изображение № 25<br>
                  <img src="src/chapter1-25.png"><br>
                   <br>
                      Запрос, приведенный на
изображении №24 возвращает количество сотрудников,
                  у которых возраст больше 30.<br>
                     Волее сложные операции агрегации
могут быть выполнены с помощью операторов
                  GROUP BY и HAVING. GROUP BY позволяет группировать данные
по определенным столбцам, а HAVING
                  позволяет фильтровать данные, на основе агрегированной
```

информации. Например:


```
<br>
                  Изображение № 26<br>
                  <img src="src/chapter1-26.png"><br>
                  <br>
                      Запрос, представленный на
изображении №26, группирует данные по отделам
                  и возвращает только те отделы, в которых количество
сотрудников больше 10.<br>
                      SQL также предлагает возможность
совмещения данных из нескольких таблиц
                  с помощью оператора JOIN. JOIN позволяет объединить
строки из двух или более таблиц на основе
                  ключевых полей. Например: <br>
                  <hr>>
                  Изображение № 27<br>
                  <img src="src/chapter1-27.png"><br>
                     В запросе на изображении №26 мы
объединяем данные из таблиц employees
                  и departments, сопоставляя их по полю department id и id
соответственно. Затем мы выбираем имена
                  сотрудников и названия отделов. <br>
                      SQL также поддерживает
подзапросы, параметризованные запросы, объединения
                  и другие мощные функции, которые могут быть применимы при
анализе данных. <br>
                      Вместе с тем, SQL не является
идеальным инструментом для всех видов
                  анализа и манипуляции с данными. В некоторых случаях
могут потребоваться дополнительные инструменты
                  или языки программирования для более сложных вычислений и
анализа данных. Кроме того, SQL является
                  стандартным языком запросов для реляционных баз данных, а
другие типы баз данных могут использовать
                  разные языки и подходы к анализу данных. <br>
                      Более глубокие представления о
SQL будут реализованы в практической части
                  работы и приложениях. <br>
                      <b>2.5. Использование BI
инструментов для анализа данных
                      и вывода результатов</b><br>
                  <br>
```

```
    ВІ инструменты являются мощным
средством для анализа данных и вывода
                   результатов, что позволяет отдельным исследователям и
организациям использовать свои данные
                   на максимально возможном уровне. Правильное использование
ВІ инструментов помогает принимать
                   обоснованные и информированные решения, а также повышать
эффективность операционной деятельности.
                   В современном мире использование ВІ инструментов
становится все более необходимым для тех, кто хочет
                   быть конкурентоспособным и успешным. <br>
                       Одной из основных целей ВІ - это
сделать данные доступными и понятными
                   для широкого круга пользователей. С помощью BI
инструментов пользователи получают возможность
                   легко анализировать большие объемы данных и создавать
аналитические отчеты и дашборды. BI системы
                   обладают мощными функциями, которые позволяют
пользователю запрашивать данные, объединять их,
                   агрегировать и визуализировать. Это позволяет
пользователям просматривать данные в реальном
                   времени, отслеживать КРІ, выявлять тренды и
прогнозировать будущие результаты. <br>
                       ВІ позволяет пользователям
создавать информативные и наглядные графики,
                   диаграммы и дашборды, которые помогают визуализировать
данные и быстро идентифицировать тренды
                   и паттерны. Это особенно полезно для менеджеров и
руководителей, которым нужно принимать оперативные
                   решения на основе данных. <br>
                       Еще одним важным аспектом ВІ
инструментов является построение прогнозов и
                   моделирование данных. С помощью алгоритмов и методов
машинного обучения ВІ инструменты позволяют
                   пользователям строить прогнозы и моделировать будущие
результаты на основе исторических данных.
                   Это позволяет принимать решения на основе точных
прогнозов и планировать свою деятельность
                   более эффективно.<br>
```

2.6. Использование HTML/CSS

для вывода результата работы

HTML и CSS являются основными базовыми технологическими инструментами

для создания веб-страниц. Команда проекта в случае наличия данных технологий в своем портфеле

всегда может представить результат своей работы в вебе.

Это может послужить альтернативой

использованию Git, так как результат будет представлен не в виде репозитория, а виде аналога

книги с навигацией.

C использованием языка разметки HTML можно легко структурировать

и организовать содержимое, создавая разделы и подразделы, а также добавляя заголовки и различные

элементы форматирования. Кроме того, HTML позволяет добавлять гиперссылки для облегчения навигации

между разделами страницы дипломной работы.

CSS в свою очередь предоставляет возможность стилизовать элементы страницы.

С помощью CSS можно изменять цвета, шрифты, размеры, добавлять иллюстрации и фоновые изображения,

что поможет сделать страницу более привлекательной и профессиональной.

Eще одним важным аспектом создание таблиц. С помощью HTML и CSS можно

создавать различные таблицы, которые помогут представить результаты исследования и обобщить

информацию.

CSS позволяет настраивать внешний вид таблиц, добавлять стили для заголовков и ячеек, а также

добавлять разделители и анимацию.

HTML и CSS также позволяют добавлять интерактивные элементы. Если расширить

технологический стек с помощью JavaScript, который можно интегрировать в HTML-код, можно создать

раскрывающиеся списки, вкладки и кнопки, которые позволят пользователю взаимодействовать с

содержимым

страницы, например, просматривать дополнительную информацию или скрывать часть содержимого по

выбору.

Важным аспектом является также адаптивный дизайн. HTML и CSS позволяют

создавать страницы, которые будут автоматически адаптироваться под различные устройства и экраны.

```
Это значит, что страница отчета будет выглядеть хорошо на
компьютере, планшете или мобильном
                    устройстве, положительно влияя на опыт пользователей. <br>
                </div>
        </div>
        </div>
    </section>
</body>
</html>
<head>
    <meta charset="UTF-8" />
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Глава 2</title>
    <link rel="stylesheet" href="style.css">
</head>
<body>
    <header>
        <div class="center">
            <div class="logo">
                <a href="index.html">На главную</a>
                <!-- <a
href="https://drive.google.com/drive/folders/1ZugqcHTbfqaNu64ULXxcGxFxfEIRDbx
z">GoogleDrive</a> -->
            </div>
            <div class="menu">
                <a href="contents.html">Содержание</a>
                <a href="introduction.html">Введение</a>
                <a href="chapter1.html">Глава I</a>
                <!-- <a href="chapter2.html">Глава II</a> -->
                <a href="conclusion.html">Заключение</a>
                <a href="appendix.html">Приложения</a>
                <a href="literature.html">Список литературы</a>
                <!-- <a
href="https://github.com/BoginskiyVV/webpageDiploma">Репозиторий Github</a>
                <a class="btn-menu" href="#">Ссылка</a> -->
            </div>
```

</div>

3.1. Парсер на языке Python с загрузкой данных в SQL формат

Первичной практической задачей при создании поисково-аналитического

механизма было принято создание на языке Python обходчика сайтов по формируемой в процессе

исполнения основного файла базе адресов. В качестве первичной базы адресов в сети интернет

был обнаружен и скачан текстовый файл с 5 млн. адресами сайтов рунета (5019692) для дальнейшей

выемки случайным образом необходимого количества строк адресов для проведения эксперимента.

В качестве поэтапного увеличения характеристик эксперимента разработанный обходчик решено

тестировать на линейке из 10, 100, 1000 случайных адресов сайтов с использованием библиотеки random.

Для простоты первичной фазы эксперимента были выбраны встроенные библиотеки requests и sqlite3

языка Python для работы с веб-страницами и SQL-форматом соответственно. Для подсчета времени

работы программы использовалась библиотека time.

 В ходе разработки программа будет создавать базу данных SiteBase

с таблицей SiteTable, в которой будут содержаться текстовые столбцы «www» и «html».

Программа открывает существующий в текущей папке файл base.txt и записывает во вновь создаваемый

файл adresses.txt необходимое количество строк, которые представляют из себя адреса веб-страниц.

Далее создается база данных и с каждой строки файла adresses.txt мы забираем адрес страницы

в сети интернет, записываем это значение в ячейку столбца «www», соединяемся со страницей

по адресу и записываем ее содержимое в соответствующую ячейку столбца «html» таблицы SiteTable

```
базы данных. В случае ошибки соединения в столбец «html»
таблицы SIteTable записывается значение
                   NULL. В процессе исполнения программы ведется подсчет
количества скачанных сайтов, количества
                   неудавшихся соединений, а также времени исполнения кода.
Когда закончатся строки файла adresses.txt
                   программа печатает строку "адреса закончились", выводит
количество скачанных сайтов, количество
                   неудачных соединений и время, затраченное на скачивание
страниц. <br>
                   <br>
                   Изображение № 27. Первичное состояние перед исполнением
программы<br>
                   <img src="src/chapter2-27.png"><br>
                   <br>
                   Изображение № 28. Код программы<br>
                   <img src="src/chapter2-28.png"><br>
                   <img src="src/chapter2-28-1.png"><br>
                   <br>
                       Запускаем код для 10 случайных
сайтов, получаем
                   результат: <br>
                   <br>
                   Изображение № 29. Результат обработки 10 случайных
адресов<br>
                   <img src="src/chapter2-29.png"><br>
                   <br>
                      Фиксируем появление в папке
проекта файлов
                   adresses.txt и SiteBase.db:<br>
                   Изображение № 30. Изменения в папке проекта после
исполнения программы<br>
                   <img src="src/chapter2-30.png"><br>
                       Запускаем программу для выборки
из 100 адресов, получаем
                   следующий результат: <br>
                   Изображение № 31. Результат обработки 100 случайных
адресов<br>
                   <img src="src/chapter2-31.png"><br>
                   <br>>
```

```
    Запускаем программу для выборки
из 1000 адресов, получаем
                   следующий результат: <br>
                   <br>
                   Изображение № 32. Результат обработки 1000 случайных
адресов<br>
                   <img src="src/chapter2-32.png"><br>
                   <br>
                       В процессе разработки для
верификации возможности дальнейшей
                   работы с полученными данными были разработаны простейшие
механизмы проверки. Для
                   начала был написан код на Python, выводящий на печать
строки «www» из таблицы
                   SiteTable базы данных:<br>
                   <br>
                   Изображение № 33. Код программы для печати содержимого
таблицы БД<br>
                   <img src="src/chapter2-33.png"><br>
                   <br>
                      После запуска кода в терминале
происходит печать содержимого
                   ячеек «www» в виде первичного html кода, с сохранением
пробелов и отступов, как это
                   было реализовано разработчиками сайтов с соблюдением
Document Object Model. <br>
                   <br>
                   Изображение № 34. Печать содержимого БД в терминале<br>
                   <img src="src/chapter2-34.png"><br>
                   <br>
                      Далее для подтверждения
возможности работы с базой данных
                   в разрезе их анализа была реализована первичная
аналитическая программа, позволяющая
                   выявлять частотность слов на всем массиве содержимого
столбцов «html» таблицы SiteTable.<br>
                      При запуске кода (изображение №
36) мы получаем CSV файл: <br>
                   <br>
                   Изображение № 35. Содержимое сформированного CSV
файла<br>
                   <img src="src/chapter2-35.png"><br>
                   <br>>
```

```
Изображение № 35. Код формирования повторений слов в CSV
файл<br>
                   <img src="src/chapter2-35-1.png"><br>
                   <br>
                       <b>3.2. Реализация на Python
многопоточной обработки данных
                       для парсинга сайтов с последующим анализом в Jupyter
Notebook</b><br>
                      Для отработки навыков парсинга
команда писала второй скрипт,
                   основным отличием которого будет многопоточность
обработки сайтов, сохранение данных
                   в CSV формате и предобработка html данных. <br>
                      Программное обеспечение:<br>
                   • IDE VSCode 1.82.2 (user setup) <br>
                   • python 3.10.11.<br>
                      Репозиторий с исходным кодом
можно будет посмотреть
                   в приложении.<br>
                   <br>>
                       Этапы написания скрипта:<br>
                   1. Открываем терминал в VSCode "ctrl + ~" :<br>
                   • Создаем виртуальное окружение: <br>
                   <span style="color: #ff6600;">python -m venv
venv</span><br>
                   <br>
                   • Активируем виртуальное окружение: <br>
                   <span style="color:</pre>
#ff6600;">venv/scripts/activate</span><br>
                   <br>
                   • Переходим в папку с виртуальным окружением: <br>
                   <span style="color: #ff6600;">cd ./venv</span><br>
                   <br>
                   В правом нижнем углу VSCode отображается: <br>
                   <span style="color: #ff6600;">venv:venv</span><br>
                   <br>>
                   2. Добавляем/обновляем необходимые файлы: <br>
                   • Обновить pip:<br>
                   <span style="color: #ff6600;">python -m pip install --
upgrade pip</span><br>
                   <br>>
                   3. Формируем файл с зависимостями: <br>
```

```
<span style="color: #ff6600;">pip freeze >
requirements.txt</span><br>
                    <br>
                       Bce актуальные библиотеки можно
посмотреть в файле
                    requirements.txt <br>>
                    <br>
                    • Установить зависимости при клонировании репозитория
локально: <br>
                    <span style="color: #ff6600;">python -m pip install -r
requirements.txt</span><br>
                    <br>
                    4. Загрузить файл с доменом ru<br>
                    Ссылка на сайт с базой зарегистрированных доменов в
приложении. <br>
                    Скачиваем zip папку и распаковываем ее в нашу папку с
проектом. <br>
                    <br>
                    <br>
                    Изображение № 36. Папка с проектом, содержащая txt файл
co списком url - адресов<br>
                    <img src="src/chapter2-36.png"><br>
                    <br>
                    Изображение № 37. Фрагмент списка с url адресами<br>
                    <img src="src/chapter2-37.png"><br>
                    <br>>
                        В последующем нам придется
дописать https://
                    ко всем строчкам в этом списке, чтобы мы могли зайти на
все эти сайты.
                    Список состоит из 5 млн адресов. Наша задача будет
состоять в том, чтобы
                    спрасить как можно больше сайтов. Пройтись по всем не
получится, так как
                    сайты в данном списке могут быть хорошо защищены от
парсинга, а могут
                    и не быть запущены на сервере. <br>
                    <hr>>
                    5. Создание скрипта в файле spider_parser.py<br>
                    Импортируем библиотеки<br>
                    <br>>
                    Изображение № 38. Импорт библиотек<br>
                    <img src="src/chapter2-38.png"><br>
                    <hr>>
```

```
• threading - это стандартная библиотека Python, которая
предоставляет
                    инструменты для многозадачности. threading позволяет
управлять потоками.
                    Потоки позволяют выполнять несколько задач одновременно,
что полезно для
                    выполнения параллельных операций. Мы воспользуемся этой
библиотекой, чтобы
                    распараллелить потоки обработки парсинга сайтов. <br>
                    • requests - это библиотека Python для выполнения HTTP-
запросов. Она
                    облегчает отправку НТТР-запросов к веб-серверам и
получение ответов.
                    requests позволяет нам взаимодействовать с веб-ресурсами,
получать данные,
                    отправлять данные и многое другое. <br>
                    • csv - это модуль Python для работы с файлами в формате
CSV (Comma-Separated Values).
                    Формат CSV используется для хранения табличных данных,
где значения разделены запятыми
                    (или другими разделителями). Модуль csv предоставляет
функциональность для чтения
                    CSV-файлов, записи данных в CSV-файлы и обработки
табличных данных. <br>
                    • re - это модуль Python для работы с регулярными
выражениями. Регулярные выражения
                    (или регулярные выражения) используются для поиска и
манипуляции текстовой информацией
                    на основе шаблонов. Модуль ге позволит нам отфильтровать
полученный текст по русским
                    символам. В итоге мы получим чистый текст с русским
символами. <br>
                    Создадать файл modif_file.py, прописать функцию, которая
будет к каждой строчке url
                    адресов добавлять https:\<br>
                    <br>
                    Изображение № 39. Функция, модифицирующая файл<br>
                    <img src="src/chapter2-39.png"><br>
                    Импортируем функцию url mod в основной файл
spider parser.py<br>
                    <br>>
                    Изображение № 40. Импорт функции<br>
                    <img src="src/chapter2-40.png"><br>
```

```
<br>
                       Чтение файла moifiled urls.txt.
При чтении мы будем
                   записывать данные в формат List и применим метод strip()
- это метод строк
                   в Python, который используется для удаления начальных и
конечных пробелов
                   (или других указанных символов) из строки. Он создаст
новую строчку без
                   лишних пробелов. <br>
                   <br>
                   Изображение № 41. Чтение url - адресов<br/>
                   <img src="src/chapter2-41.png"><br>
                   <hr>>
                       Прописать функцию run, которая
будет отвечать
                   за парсинг, считывание html кода с первых страниц сайтов.
Можно совершенствовать
                   код, до "пробегания" по всем страницам сайтов, но так как
это только начало проекта,
                   и мы практикуем основные навыки работы с большими
данными, остановимся на самом
                   простом парсинге.<br>
                   <hr>>
                   Изображение № 42. Class Parser<br>
                   <img src="src/chapter2-42.png"><br>
                   <br>
                      C экземпляром потока и его
методами, можно ознакомиться
                   в Приложении №1. Немного модифицировали код для наших
целей и получили основную
                   часть кода, которая отвечает за парсинг сайтов. Это
многопоточный метод поиска.
                   Для нас он наиболее приемлем, так как большие объемы url
адресов обходить
                   получается намного быстрее нежели при обычном обходе. <br>
                   <br>>
                   Изображение № 43. Class Parser<br>
                   <img src="src/chapter2-43.png"><br>
                   В конце мы создадим строку заголовка для csv файла. <br>
                   <br>
                   Изображение № 44. Запись CSV-файла<br>
                   <img src="src/chapter2-44.png"><br>
```

```
<br>
                   Проблемы, с которыми мы
столкнулись: <br>
                    1. Не все сайты из скачанного
списка существуют,
                возможно сервер не работает; <br>
                    2. Не все сайты позволяют
обходчику прочитать данные; <br>
                    3. Сохранить данные такого объема
на GitHub обычным
                способом нельзя. <br>
                   Решение проблем: <br>
                    1. Решили продолжать работу по
парсингу по тем сайтам,
                которые существуют. Список сузится, но для начала нашего
проекта и этого достаточно. <br>
                    2. Решение ошибки 443 находится
на стадии разработки; <br>
                    3. Использовали метода работы с
большими данными, подключив
                GIT LFS;<br>
                    4. Указали файлы отслеживания,
сохранили изменения
                и перенесли на Git Hub. <br>
                <br>
                Изображение № 45. Git LFS<br>
                <img src="src/chapter2-45.png"><br>
                <hr>>
                    Промежуточные результаты: <br>
                    1. Написана программа для
многопоточного парсинга
                с извлечением данных с первых страниц сайтов и
сохранением их в CSV формате. <br>
                    2. Количественные данные:<br>
                    - Из базы в 5 млн сайтов<br>
                     - Обработано 100 000 сайтов<br/>сbr>
                      - Спарсено 11 000 сайтов <br>
                <br>>
                   Aнализ полученных данных и
Jupyter Notebook<br>
                   Данные из файла parser.csv
представлены двумя столбцами,
                в первом стоблце url адреса, во втором столбце спарсенные
данные в виде текста.
```

```
Для анализа данных будем использовать Anaconda 1.12.1,
python 3.10.13
                  (версии выше не поддерживают библиотеку nltk). <br>
                      Загрузку файла осуществили с
помощью pandas, подсчет
                  строк провели с помощью мультипроцессинга, чтобы
закрепить навык работы с большими
                  данными. Количество строк 11372. Удалили строчки
содержащие строчки NaN, осталось
                  9919 строк. Определим функцию, которая очищает текст от
некорректных символов.
                  Применили данную функцию к нашему датафрейму, используя
два способа: последовательная
                  обработка и мультипроцессинг. <br>
                     Aнализ текста: получим число слов
в тексте и число уникальных
                  слов с помощью модуля collections. Результат «Число
уникальных слов: 8991», «Сумма
                  частот уникальных слов: 9919». <br>
                     Поиск популярных слов
производится с помощью библиотеки
                  NLTK. Вывод представлен в виде кортежа, таблицы, графика
и облака слов.
                      Стемпинг и лемматизацию провести
не получилось, так как
                  получили ошибку «MemoryError: ». Связано с большим
объемом данных, попробуем
                  в дальнейшем использовать PySpark. <br>
                     Частотный анализ слов и
словосочетаний провели по тем позициям,
                  которые нас интересуют для дальнейшего исследования. <br>
                      Ознакомиться с подробным анализом
в jupyter notebook можно
                  в приложении. <br>
                     Peзультаты:<br>
                      1. Провели анализ текста и
обработку естественного языка
                  с помощью библиотеки NLTK; <br>
                      2. Освоили методы работы с
большими данными
                  в jupyter notebook.<br>
                  <br>>
                  Изображение № 46. Схема анализа<br>
                  <img src="src/chapter2-46.png"><br>
```

```
<br>
                       <b>3.3 Анализ полученных данных
средствами SQL</b><br>
                      Возьмем нашу базу, полученную при
помощи формирования данных
                   языком Phyton в CSV - файл и преобразуем для чтения с
помощью SQL. При помощи программы
                   EXCEL подготовим данные для таблицы. <br>
                   <br>
                   Изображение № 47. Подготовка данных<br>
                   <img src="src/chapter2-47.png"><br>
                   <br>
                      Данные необходимо сохранить в
формате :65001: Юникод (UTF-8),
                   указать на формат данных: с разделителем. Вы можете
начать импорт с любой строки.
                   Указать заголовки - выбрав галочкой «Мои данные содержат
заголовки»
                   (См. Изображение № 48).<br>
                   <hr>>
                   Изображение № 48.<br>
                   <img src="src/chapter2-48.png"><br>
                   <br>
                       Нажав кнопку далее, определим
разделители. В нашем случае,
                   это «знак табуляции» и «пробел» (См. Изображение № 49). И
кнопку считать
                   последовательные разделители одним. Выставляем
ограничение строк до 100.<br>
                   <hr>>
                   Изображение № 49.<br>
                   <img src="src/chapter2-49.png"><br>
                   <br>
                   При количестве строк величиной 100 получается 15100 -
ячеек. <br>
                   <hr>>
                   Изображение № 50.<br>
                   <img src="src/chapter2-50.png"><br>
                   Сохраняем файл в формате CSV (разделитель- запятая). <br>
                   <br>>
                   Изображение № 51.<br>
                   <img src="src/chapter2-51.png"><br>
                   <br>
```

```
В файле закаченной базы данных 12924 строк. Разбираем
данные по файлам поскольку
                    с большим объемом в 500 строк и в 75500 ячеек MySQL не
справился.<br>
                    Изображение № 52.<br>
                    <img src="src/chapter2-52.png"><br>
                    <br>
                    Изображение № 53.<br>
                    <img src="src/chapter2-53.png"><br>
                    <br>
                    12924 строки делим по 150 строк в файл и получаем 85
документов для анализа.
                    Заходим в MYSQL в базе данных необходимо выбрать
кодировки согласно рисунку 8. <br>
                    <br>
                    Изображение № 54.<br>
                    <img src="src/chapter2-54.png"><br>
                    <br>
                    Нажимаем правой кнопкой мыши на вкладке справа. <br/>
                    <br>>
                    Изображение № 55.<br>
                    <img src="src/chapter2-55.png"><br>
                    «Table Data Import Wizart» Появляется табло скачивания
файла.<br>
                    <br>
                    Изображение № 56.<br>
                    <img src="src/chapter2-56.png"><br>
                    <br>>
                    Нажимаем кнопку «NEXT». Появляется окно с символами из
нашего файла.<br>
                    Изображение № 57.<br>
                    <img src="src/chapter2-57.png"><br>
                    <br>
                    Нажимаем кнопку «NEXT». <br>
                    <br>
                    Изображение № 58.<br>
                    <img src="src/chapter2-58.png"><br>
                    Таблица появилась в окне «OUTput». Меняем Имя второй
колонки, поскольку
                    наличие синтаксических символов мешают
программированию. <br>
```

```
Изображение № 59.<br>
                   <img src="src/chapter2-59.png"><br>
                   <br>
                   Ищем совпадения для построения карты рынка товаров
                   и услуг (Изображения № 60 и 61).<br>
                   <br>
                   Изображение № 60.<br>
                   <img src="src/chapter2-60.png"><br>
                   <br>
                   Изображение № 61.<br>
                   <img src="src/chapter2-61.png"><br>
                   Изображение № 62.<br>
                   <img src="src/chapter2-62.png"><br>
                   <br>
                   С физическими адресами обнаружено 18 объектов. <br>
                   Результат: <br>
                   В результате анализа базы данных 85 файлов обнаружено
1748
                   объектов с товарами и услугами и 1386 физических
адресов. <br>
                   <br>
                       <b>3.4. Анализ и вывод полученных
данных с
                       помощью инструментов BI</b><br>
                       В Power BI можно использовать
функции и инструменты
                   для поиска строк с повторяющимися значениями в строке.
Одной из возможностей
                   является использование Power Query, чтобы выполнить эту
задачу. Вот шаги,
                   которые были реализованы: <br>
                       Открываем Power Query Editor,
выбрав соответствующую
                   опцию в меню Power BI. «Текстовы или CSV-файл»: <br/>
                   <br>
                   Изображение № 62.<br>
                   <img src="src/chapter2-62-1.png"><br>
                   Загружаем или создаем таблицу, в которой будем искать
строки с повторяющимися
                   значениями.<br>
                   <hr>>
```



```
Изображение № 63.<br>
                   <img src="src/chapter2-63.png"><br>
                   <br>
                      Поскольку нам необходимо найти
совпадение в тексте
                   в 2 столбца: URL и content, нажимаем «Извлечение таблицы
с использованием
                   примеров.»<br>
                   <br>
                   Изображение № 64.<br>
                   <img src="src/chapter2-64.png"><br>
                   <br>
                       Вставляем в заголовок столбец
URL, нажимаем на «+»,
                   во второй столбец пишем «content» <br>
                       Когда вид столбцов организован,
выставляем на
                   основе 200 строк в окне, разделитель - точка с запятой. -
извлекаем. <br>
                   <hr>>
                   Изображение № 65.<br>
                   <img src="src/chapter2-65.png"><br>
                   <br>
                   Появляется таблица с колонками, выделяем столбец,
содержащий строковые
                   значения, в которых нужно найти повторы. <br>
                   <br>
                   Изображение № 66.<br>
                   <img src="src/chapter2-66.png"><br>
                   <br>
                   Нажимаем в верхнем меню во вкладке «Преобразование»
кнопку «Извлечение»,
                   «Текст после разделителя». <br>
                   <br>>
                   Изображение № 67.<br>
                   <img src="src/chapter2-67.png"><br>
                   <br>
                   Вводим текст в нашем случае это «адрес», «товар»,
«услуги». С формированием
                   одноименных столбцов. <br>
                   <br>
                   Изображение № 68.<br>
                   <img src="src/chapter2-68.png"><br>
                   <br>
```

```
Далее выводим результат после удаления пустых строк. <br>
                 <br>
                 Изображение № 68.<br>
                 <img src="src/chapter2-68-1.png"><br>
                 <br>
                 Результат:
                 В поле адресов - 831 отдельных и 791 уникальных, товаров
- 232 отдельных
                 и 222 уникальных, услуг - 225отдельных и 219
уникальных. <br>
                 <br>
                     <b>3.5 Разработка аналитического
парсер-бота
                     с помощью Python</b><br>
                     Этот раздел представляет
разработку простого телеграм-бота,
                 который выполняет анализ веб-сайтов, веденных
пользователем. Давайте рассмотрим
                 его разработку и функциональность по шагам: <br>
                 <hr>>
                 Изображение № 69.<br>
                 <img src="src/chapter2-69.png"><br>
                 <br>
                     Импортированные библиотеки:<br>
                      - `telebot`: используется для
работы с Telegram API
                 и создания бота. <br>
                     - `requests`: используется для
отправки НТТР-запросов
                 к веб-сайту и получения его содержимого. <br>
                    - `BeautifulSoup`: используется
для парсинга
                 HTML-контента и извлечения текста. <br>
                      - `Counter`: используется для
подсчета количества
                 встречающихся элементов в последовательности. <br>
                 <br>
                 Изображение № 70.<br>
                 <img src="src/chapter2-70.png"><br>
                    Установка токена бота и
инициализация бота:<br>
                     - `TOKEN`: хранит токен бота. <br>
```

```
    - `bot = telebot.TeleBot(TOKEN)`:
Создает экземпляр бота
                 с использованием указанного токена. <br>
                 <br>
                 Изображение № 71.<br>
                 <img src="src/chapter2-71.png"><br>
                 <br>
                     Обработчики сообщений:<br>
                     anbsp; - `handle_start(message)`:
Обрабатывает команду `/start`.
                 Отправляет приветственное сообщение и запрашивает адрес
веб-сайта.<br>
                     -
`handle website input(message)`: обрабатывает
                 введенный пользователем адрес веб-сайта. Получает
содержимое веб-сайта
                 и предоставляет пользователю опции для анализа. <br>
                     -
`handle_analysis_choice(message, html_content)`:
                 Обрабатывает выбор пользователя по типу анализа (топ-10
символов, топ-10 слов,
                 количество символов, пробелов и самый частый символ,
полный анализ).<br>
                 <hr>>
                 Изображение № 72.<br>
                 <img src="src/chapter2-72.png"><br>
                 <br>
                    Выбор анализа:<br>
                    - "1. Топ-10 символов": находит и
выводит 10 самых часто
                 встречающихся буквенных символов в тексте. <br>
                     - "2. Топ-10 слов": находит и
выводит 10 самых часто
                 встречающихся слов (более 5 символов) в тексте. <br>
                     - "3. Количество символов,
пробелов, самый частый символ":
                 выводит общее количество символов и пробелов, а также
самый часто встречающийся
                 символ и его количество в тексте. <br>
                     - "4. Полный анализ": Выполнение
всех предыдущих пунктов. <br>
                 <br>>
                    Запуск бота:<br>
```

```
    anbsp; - `bot.polling(none_stop=True)`:
Запускает бота, который
                  непрерывно опрашивает серверы Telegram на предмет новых
сообщений. <br>
                  Изображение № 73.<br>
                  <img src="src/chapter2-73.png"><br>
                  <br>
                  Пример работы бота с сайтом google.com<br>
                  Изображение № 74.<br>
                  <img src="src/chapter2-74.png"><br>
                  <br>
                      <b>3.5. Вывод результатов проекта
с помощью создания
                      сайта на HTML/CSS</b><br>
                     В целях реализации рекомендаций
по написанию дипломной работы,
                  обобщения всех материалов коллективной дипломной работы и
визуализации результатов
                  проекта командой было принято решение о создание простой
версии веб-страница.
                  В качестве шаблона сайта был выбран один из вариантов
простого сайта, а именно шаблон
                  был взят с репозитория github.com
(https://github.com/guih58/OnePage).<br>
                      Style.css позволяет визуально
оформить страницу раскрасить
                  подзаголовки, поменять фон или отформатировать
изображение. В каталоге scr/ будут
                  размещены снимки экрана, которые в последствии будут
отображены на страницах.<br>
                     С учётом множества различных
материалов логика сайта была
                  построена таким образом, чтобы пользователь мог
просмотреть вложенный контент
                  и переключаться между различными частями дипломной
работы, используя меню,
                  расположенное в верхней части экрана. <br>
                  <br>>
                  Изображение № 75.<br>
                  <img src="src/chapter2-75.png"><br>
                   <br>
                     Для создания и визуализации
логического дерева был использован
```

```
онлайн pecypc https://tree.nathanfriend.io/.<br>
                     В качестве IDE использовалось
программное средство
                  Visual Studio Code. Формирование верхнего меню
осуществляется в тегах "body header". <br>
                  <br>
                  Изображение № 76.<br>
                  <img src="src/chapter2-76.png"><br>
                      Основной текст страниц также
размещен в тегах "body header".
                  Для примера взят фрагмент кода страницы contents.html,
входящей в состав страниц сайта. <br>
                  <hr>>
                  Изображение № 77.<br>
                  <img src="src/chapter2-77.png"><br>
                  <br>
                      Конечным результатом написания
кода является отображение
                  информации в веб браузере. <br>
                  <br>
                  Изображение № 78.<br>
                  <img src="src/chapter2-78.png"><br>
                      Полный код сайта находиться в
репозитории github.com,
                  а также сохранён и доступен на drive.google.com.<br>
               </div>
       </div>
       </div>
   </section>
</body>
</html>
<head>
   <meta charset="UTF-8" />
```

```
<meta http-equiv="X-UA-Compatible" content="IE=edge">
   <meta name="viewport" content="width=device-width, initial-scale=1.0">
   <title>Заключение</title>
   <link rel="stylesheet" href="style.css">
</head>
<body>
   <header>
       <div class="center">
           <div class="logo">
               <a href="index.html">На главную</a>
               <!-- <a
href="https://drive.google.com/drive/folders/1ZugqcHTbfqaNu64ULXxcGxFxfEIRDbx
z">GoogleDrive</a> -->
           </div>
           <div class="menu">
               <a href="contents.html">Содержание</a>
               <a href="introduction.html">Введение</a>
               <a href="chapter1.html">Глава I</a>
               <a href="chapter2.html">Глава II</a>
               <!-- <a href="conclusion.html">Заключение</a> -->
               <a href="appendix.html">Приложения</a>
               <a href="literature.html">Список литературы</a>
               <!-- <a
href="https://github.com/BoginskiyVV/webpageDiploma">Репозиторий Github</a>
               <a class="btn-menu" href="#">Ссылка</a> -->
           </div>
       </div>
       <!--Center-->
   </header>
</header>
<section class="main">
   <div class="center">
       <div class="main cta">
           <h2>3аключение</h2>
           <br>
               -->
               <br>
                  В представленной коллективной
дипломной работе
               команда проекта ставила перед собой задачи отразить
теоретические знания
```

и практический опыт, полученные в результате обучения

аналитическим

специализациям на платформе GeekBrains. Накопленные знания и

опыт были

применены для решения задач поиска информации в сети Интернет

с последующим

хранением и анализом данной информации.

Ocновными компонентами

технологического стека,

представленного GeekBrains в рамках курсов аналитических специальностей,

по мнению членов команды, являются языки Python и SQL, позволяющие решать

широкий спектр аналитических задач и разрабатывать как небольшие самостоятельные

инструменты, так и целые аналитические комплексы. Также отличным инструментарием

для анализа и отображения данных является ВІ, кратко разобранный в теоретической

части. Наибольшее внимание в главе, посвященной теории, было уделено различным

библиотекам языка Python, которые позволяют собирать информацию в сети Интернет

и анализировать полученные данные. Для библиотек requests, BeautifulSoup, Scrapy,

Selenium, Pandas, Matplotlib были даны характеристики и описаны особенности

использования данных инструментов. Также в главе представлено описание использования

различных форматов хранения полученных данных, таких как CSV и JSON, проведено

сравнение использования данных инструментов, в том числе относительно использования

SQL-форматов.

B практической части команда попыталась реализовать различные

варианты создания парсинговых механизмов и применить несколько подходов для анализа

полученных данных. В связи с тем, что проект изначально реализовывался на отдельных

локальных машинах, было принято решение ограничиться работой только с текстовой информацией,

на заходя в область больших данных. Изначальный файл с адресами для программы-обходчика содержал более 5 млн. адресов сайтов Рунета, которые и стали источником текстовых данных

для практической работы. В первом и самом простом варианте реализации парсера, отраженном

в разделе 3.1., текстовое (html) содержимое сайтов полностью забиралось на хранение

в ячейку таблицы базы данных SQL. Такой способ хорошо работает в случае больших данных

и хороших скоростей каналов, когда необходимо собрать исчерпывающую информацию. Среднее

время на скачивание html содержимого одного сайта составило более 6 секунд. Это четко

показывает, что для решения поисковых задач на больших массивах адресов сайтов метод

работает достаточно медленно и лучше оптимизировать задачу уже в процессе обхода, экономя

аппаратные ресурсы.

Bторой подход к парсеру сайтов на Python был более осознанным

и углубленным, поскольку имело место применение многопоточного подхода и предварительной

очистки данных, собираемых для дальнейшего анализа. Также стоит отметить, для хранения

данных был использован формат CSV, что позволило увеличить масштаб работы

поисково-аналитического механизма с ранее использованного объема обхода в тысячу

страниц до десятков тысяч обрабатываемых сайтов. В Приложении №1 представлен Jupyter

Notebook проекта, который дает полное понимание о подходах, примененных при решении

задач.

Также команда проекта показала на практике использование SQL

и ВІ инструментов для анализа данных, были подтверждены компетенции разработки

чат-ботов на примере создания парсинг-бота, производящего анализ сайта, адрес которого

введен пользователем. В качестве инструмента отображения результатов дипломной работы

командой был разработан сайт проекта на технологии HTML/CSS с прицелом на будущее

создание интерактивного инструмента отображения результатов анализа данных.<br

```
   В перспективе проект будет реализован
командой уже не на
                локальных машинах. Планируется разворачивание поисково-
аналитического механизма
                на внешних серверах, что придаст проекту совершенно иные
возможности. Также в перспективе
                будут задействованы механизмы обработки, хранения и анализа
больших данных, что потребует
                от команды вложения в проект денежных средств, что должно
быть обусловлено практической
                обоснованностью проекта в коммерческом плане.
                <br>
            </div>
    </div>
    </div>
</section>
</body>
</html>
<head>
    <meta charset="UTF-8" />
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Список литературы</title>
    <link rel="stylesheet" href="style.css">
</head>
<body>
    <header>
        <div class="center">
            <div class="logo">
                <a href="index.html">На главную</a>
                <!-- <a
href="https://drive.google.com/drive/folders/1ZugqcHTbfqaNu64ULXxcGxFxfEIRDbx
z">GoogleDrive</a> -->
            </div>
            <div class="menu">
                <a href="contents.html">Содержание</a>
                <a href="introduction.html">Введение</a>
                <a href="chapter1.html">Глава I</a>
                <a href="chapter2.html">Глава II</a>
```

```
<a href="conclusion.html">Заключение</a>
                <a href="appendix.html">Приложения</a>
                <!-- <a href="literature.html">Список литературы</a> -->
                <!-- <a
href="https://github.com/BoginskiyVV/webpageDiploma">Репозиторий Github</a>
                <a class="btn-menu" href="#">Ссылка</a> -->
            </div>
        </div>
        <!--Center-->
    </header>
</header>
<section class="main">
    <div class="center">
        <div class="main cta">
            <h2>Список использованной литературы</h2>
            <br>
                <!-- &nbsp;&nbsp;&nbsp;&nbsp;&nbsp;<b>Заключение</b><br><br></br></r>
-->
                <hr>>
                1. Беляева И. В. Архитектура информационных
                систем: учебное пособие / И. В. Беляева. - Ульяновск: УлГТУ,
                2019. 192 c.<br>
                2. Митчелл Райан. Современный скрапинг
                веб-сайтов с помощью Python. Санкт-Петербург: Издательство
                «Питер», 2021. 497 c.<br>
                3. Равив Г. Power Query в Excel и Power
                ВІ / Г. Равив. — Санкт-Петербург: БХВ, 2021. 480 с. <br>
                4. База зарегистрированных доменов разбитая
                по зонам [Электронный ресурс].
                URL: https://purecrawl.com/download/domains
                (дата обращения: 12.10.2023). <br>
                5. Базы данных. Практическое применение
                СУБД SQL и NoSQL-типа для проектирования информационных
систем:
                учеб. пособие / С.А. Мартишин, В.Л. Симонов, М.В. Храпченко.
                — Москва: «ФОРУМ», 2018. 368 с. [Электронный ресурс].
                URL: http://znanium.com/ catalog/product/926871/(дата
обращения:
                23.09.2023).<br>
                6. Библиотека Pandas // ru.wikipedia.org
                [Электронный ресурс]. URL: ru.wikipedia.org/wiki/Pandas
                (дата обращения: 12.10.2023). <br>
                7. Библиотека Requests // python.ru
```

```
[Электронный ресурс]. URL: python.ru/post/97/ (дата
обращения:
                12.10.2023).<br>
                8. Библиотека Python для извлечения данных
                из файлов HTML и XML [Электронный ресурс].
                URL: https://www.crummy.com/software/ BeautifulSoup/bs4/doc/
                (дата обращения: 25.09.2023) <br>
                9. Гуриков, С. Р. Основы алгоритмизации
                и программирования на Python : учеб. пособие /
                С.Р. Гуриков. — Москва: «ФОРУМ», 2018. 343 с. [Электронный
pecypc].
                URL: https://znanium.com/catalog/product/924699 (дата
обращения:
                20.01.2020).<br>
                10. Документация по BeautifulSoup
                // wiki.python.su [Электронный ресурс]. URL: wiki.python.su
                (дата обращения: 12.10.2023).<br>
                11. Краткое руководство по веб-парсингу
                [Электронный ресурс].
                URL: https://habr.com/ru/companies/selectel/articles/754674/
                (дата обращения: 18.10.2023) <br>
                12. Метод загрузки больших файлов GitHub
                [Электронный ресурс].
                URL: https://all-
python.ru/osnovy/threading.html?ysclid=lmxayom1wc333926930
                (дата обращения: 10.10.2023). <br>
                13. Миронов М. «Парсинг»: Лучшие программы для
                парсинга / М. Миронов //iDatica.com: [Электронный ресурс].
                URL: https://idatica.com/blog/programmy-dlya-parsinga-
dannykh-v-2020-godu
                /(дата обращения: 25.09.2023)<br>
                14. Питинов А. «Парсер простыми словами»: как его
                настроить и пользоваться / A. Питинов // semantica.in:
[Электронный ресурс].
                URL: https:// semantica.in/blog/chto-takoe-parser.html
                (дата обращения: 02.10.2023) <br>
                15. Парсинг Json в Python 3 // all-python.ru
                [Электронный ресурс]. URL: all-python.ru/osnovy/json.html
(дата обращения:
                12.10.2023).<br>
                16. Руководство разработчика MySQL Connector/NET
                [Электронный ресурс]. URL:
https://dev.mysql.com/doc/connector-net/en/
                (дата обращения 16.10.2023) <br>
```

```
17. Северянин С. «Парсинг»: Что такое парсер и как
                он работает / С. Северянин // timeweb.ru: [Электронный
pecypc].
                URL: https:// timeweb.com/ru/community/articles/chto-takoe-
parser
                (дата обращения: 04.10.2023) <br>
                18. Pandas – библиотека Python [Электронный ресурс].
                URL: https:// blog.skillfactory.ru/glossary/pandas/
                /(дата обращения: 05.10.2023)<br>
                19. Pandas [Электронный ресурс].
                URL: https://pandas.pydata.org/ getting_started.html/
                (дата обращения: 05.10.2023) <br>
                20. Parser [Электронный ресурс].
                URL: https://github.com /FiryuzaLapteva/Parser
                (дата обращения: 15.10.2023) <br>
                21. SQL или NoSQL [Электронный ресурс].
                URL: https://habr.com/ru /company/ruvds/blog/324936/ (дата
обращения: 07.10.2023).
                Threading в Python 3: работа с потоками [Электронный ресурс].
                URL: https://all-
python.ru/osnovy/threading.html?ysclid=lmxayom1wc333926930
                (дата обращения: 11.10.2023).<br>
                22. Threading в Python 3: работа с потоками
                [Электронный ресурс].
                URL: https://all-
python.ru/osnovy/threading.html?ysclid=lmxayom1wc333926930
                (дата обращения: 11.10.2023).<br>
            </div>
    </div>
    </div>
</section>
</body>
</html>
```