**ТЕХНИЧЕСКАЯ ДОКУМЕНТАЦИЯ**

*Для всех членов команды предоставляется доступ в закрытый GitHub репозиторий. Для получения доступа обратитесь к старшему разработчику.*

**1. Бекенд**

**1.1. Парсинг**

**1.1.1. Возможности**

Предоставлены следующие возможности для парсинга (веб-скрейпинга):

* парсинг списка компаний с информацией о каждой компании;
* парсинг статей из блога компании
* парсинг списка компаний с информацией о каждой компании с парсингом статей из блога каждой компании.

*У вас должен быть установлен* ***Google Chrome*** *версии* ***111.0.5563.65*** *или выше. Скачайте и разместите* ***googleDrive****r для соответвующей версии Google Chrome в корне рядом с модулями для парсинга:* ***parserCompany.py****,* ***parserArticles.py****,* ***ultraParser.py****.*

**1.1.2. Устройсво парсера**

Некоторые данные на сайте генерируются динамически при помощи JavaScript. В таком случае получить эти данные силами Beautiful Soup не удастся.

Beautiful Soup исследует исходный код страницы, не исполняя JavaScript. Для исполнения JavaScript кода и получения страницы, идентичной той, которую видно в браузере, можно использовать Selenium WebDriver - это набор драйверов для различных браузеров, снабжающийся библиотеками для работы с этими драйверами.

Основным инструментом будет являться **Selenium**.

В данном случае будут использоваться следующие локаторы Selenium: **By.XPATH**, **By.CLASS\_NAME**.

**1.1.3. Модули**

**1.1.3.1. parserCompany**

Предоставляет класс **Parser** для осуществления парсинга списка компаний с информацией о каждой компании.

Начало работы:

* Импортируйте класс Parser из модуля companyParser:

`from companyParser import Parser`

* Создайте экземпляр импортированного класса:

`companyParser = Parser()`

* Запустите парсер:

` companyParser.start()`

**1.1.2.1. Используемые локаторы**

*Таблица 1. Используемые локаторы для модуля parserCompany*

|  |  |  |
| --- | --- | --- |
| Название поля | Используемый селектор | Технология |
| Блок компаний | "tm-companies" | Beautiful Soup |
| Список всех компаний | "tm-companies\_\_item tm-companies\_\_item\_inlined" | Beautiful Soup |
| Блок краткой информации о кампании | f"//body/div[@id='app']/div[1]/div[2]/main[1]/div[1]/div[1]/div[1]/div[1]/div[1]/div[3]/div[2]/div[{companyID}]/div[1]/div[1]/div[1]" | Selenium |
| Название компании | "tm-company-snippet\_\_title" | Selenium |
| Описание компании | "tm-company-snippet\_\_description" | Selenium |
| Ссылка на профиль компании | "tm-company-snippet\_\_title" | Selenium |
| Блок на цифровые показатели компании | f"//body/div[@id='app']/div[1]/div[2]/main[1]/div[1]/div[1]/div[1]/div[1]/div[1]/div[3]/div[2]/div[{companyID}]/div[2]" | Selenium |
| Рейтинг компании | f"/html[1]/body[1]/div[1]/div[1]/div[2]/main[1]/div[1]/div[1]/div[1]/div[1]/div[1]/div[3]/div[2]/div[{companyID}]/div[2]/span[1]" | Selenium |
| Количество подписчиков компании | f"/html[1]/body[1]/div[1]/div[1]/div[2]/main[1]/div[1]/div[1]/div[1]/div[1]/div[1]/div[3]/div[2]/div[{companyID}]/div[2]/span[2]" | Selenium |
| Блок хабов компании | f"//body/div[@id='app']/div[1]/div[2]/main[1]/div[1]/div[1]/div[1]/div[1]/div[1]/div[3]/div[2]/div[{companyID}]/div[1]/div[2]" | Selenium |
| Хабы коспании | "tm-companies\_\_hubs-item" | Selenium |
| Блок отраслей компании | "tm-company-profile\_\_categories" | Selenium |
| Отрасли компании | "tm-company-profile\_\_categories-wrapper" | Selenium |
| О компании | "//body/div[@id='app']/div[1]/div[2]/main[1]/div[1]/div[1]/div[2]/div[1]/div[1]/div[2]/section[1]/div[1]/div[1]/dl[2]/dd[1]/span[1]""//body/div[@id='app']/div[1]/div[2]/main[1]/div[1]/div[1]/div[2]/div[1]/div[1]/div[2]/section[1]/div[1]/div[1]/dl[3]/dd[1]/span[1]" | Selenium |

**1.1.3.2. parserArticles**

Предоставляет класс **ArticleParser** для осуществления парсинга списка статей компании.

Начало работы:

* Импортируйте класс ArticleParser из модуля parserArticles:

`from companyParser import Parser`

* Создайте экземпляр импортированного класса:

`articleParser = Parser()`

* Запустите парсер:

` articleParser.start()`

**1.1.2.1. Используемые локаторы**

*Таблица 2. Используемые локаторы для модуля parserArticles*

|  |  |  |
| --- | --- | --- |
| Название поля | Используемый селектор | Технология |
| Список статей на странице | "tm-articles-list\_\_item" | Beautiful Soup |
| Заголовок статьи | "tm-article-snippet\_\_title-link" | Beautiful Soup |
| Дата публикации | "time"[title] | Beautiful Soup |
| Блок краткой информации о кампании | f"//body/div[@id='app']/div[1]/div[2]/main[1]/div[1]/div[1]/div[1]/div[1]/div[1]/div[3]/div[2]/div[{companyID}]/div[1]/div[1]/div[1]" | Selenium |
| Название компании | "tm-company-snippet\_\_title" | Selenium |
| Описание компании | "tm-company-snippet\_\_description" | Selenium |
| Ссылка на профиль компании | "tm-company-snippet\_\_title" | Selenium |
| Блок на цифровые показатели компании | f"//body/div[@id='app']/div[1]/div[2]/main[1]/div[1]/div[1]/div[1]/div[1]/div[1]/div[3]/div[2]/div[{companyID}]/div[2]" | Selenium |
| Рейтинг компании | f"/html[1]/body[1]/div[1]/div[1]/div[2]/main[1]/div[1]/div[1]/div[1]/div[1]/div[1]/div[3]/div[2]/div[{companyID}]/div[2]/span[1]" | Selenium |
| Количество подписчиков компании | f"/html[1]/body[1]/div[1]/div[1]/div[2]/main[1]/div[1]/div[1]/div[1]/div[1]/div[1]/div[3]/div[2]/div[{companyID}]/div[2]/span[2]" | Selenium |
| Блок хабов компании | f"//body/div[@id='app']/div[1]/div[2]/main[1]/div[1]/div[1]/div[1]/div[1]/div[1]/div[3]/div[2]/div[{companyID}]/div[1]/div[2]" | Selenium |
| Хабы коспании | "tm-companies\_\_hubs-item" | Selenium |
| Блок отраслей компании | "tm-company-profile\_\_categories" | Selenium |
| Отрасли компании | "tm-company-profile\_\_categories-wrapper" | Selenium |
| О компании | "//body/div[@id='app']/div[1]/div[2]/main[1]/div[1]/div[1]/div[2]/div[1]/div[1]/div[2]/section[1]/div[1]/div[1]/dl[2]/dd[1]/span[1]""//body/div[@id='app']/div[1]/div[2]/main[1]/div[1]/div[1]/div[2]/div[1]/div[1]/div[2]/section[1]/div[1]/div[1]/dl[3]/dd[1]/span[1]" | Selenium |

**1.1.3.3. ultraParser**

*Данный модуль объединяет в себе функционал двух предыдущих модулей для удобства.*

Предоставляет класс **UltraParser** для осуществления парсинга списка компаний со статьями каждой компании.

Начало работы:

* Импортируйте класс UltraParser из модуля ultraParser:

`from ultraParser import UltraParser`

* Создайте экземпляр импортированного класса:

`ultraParser = UltraParser()`

* Запустите парсер:

` ultraParser.start()`

**1.2. Предобработка данных**

Предварительная обработка данных является важным шагом в процессе интеллектуального анализа данных.

**1.2.1. Модули**

**1.2.1.1. preprocessing**

Предоставляет класс **Preprocessing** для базовой предобработки данных

**1.2.1.1.1. Метод lemmatization.**

Метод ожидает аргумент **text: str**

Процесс предобработки состоит из этапов:

\* выделение токенов ("разбитие текста" на список строк, представляющих собой слова);

\* удаление пунктуации;

\* удаление эмодзи (метод **removeEmoji**);

\* полное удаление чисел;

\* лемматизация слов - процесс приведения словоформы к лемме — её нормальной форме;

\* удаление стоп-слов (слова-вредители / мусор).

Возвращает список токенов предобработанного текста **tokens: List[str, ...]**

**1.2.1.1.2. getNeedTokens**

Данный метод позволяет отобрать только нужные теги.

На вход принимает аргумент **text: List[str, ...]**

Возвращает список кортежей **words: List[Tuple[str, str]]**

**1.2.1.1.2.1. Проблема**

С русским языком есть явные проблемы, а имменно неправильное тегирование слов.

**1.2.1.1.2.2. Решение**

Создание и обучение собственного тегеровщика.

**1.2.1.1.3. getWordsFromListOfTaggedWords**

Позволяет отобрать слова от тегов.

На вход принимает аргумент **taggedWords: List[Tuple[str, str]]**

Возвращает список строк **words: List[str, ...]**

**1.2.1.1.4. removeEmoji**

Позволяет очистить текст от эмодзи.

На вход принимает аргумент **text: str**

Возвращает обработанный текст **text: str**

**1.2.1.1.5. buildWordCloudComparisonChart**

**WordCloud** - это визуальное представление списка категорий Обычно используется для описания тегов на веб-сайтах или для представления неформатированного текста.

Для каждой компании строится график сравнения: WordCloud до удаления стоп-слов и WordCloud после удаления стоп-слов.

Эти графики помогут оценить выявление стоп-слов, что очень сильно улучшит качество текста.

На вход принимает агрументы:

* **companyName: str**
* **text: str**
* **save=False: bool**

**companyName -** название компании

**text -** текст о компании

**bool** - флаг. Если **True** сохранить график, **False** вывести график на экран

*График для каждой компании можно найти папке в "****data/WordCloud****s".*

**1.3. Алгоритм обучения модели**

**1.3.1. Векторизация текста**

Для того чтобы к тексту можно было применить алгоритмы машинного обучения, необходимо провести векторизацию текста - процесс перевода (конвертация) текста в числовой формат.

Это один из ключевых этапов работы в области NLP.

**1.3.1.1. проблема**

В данный момент у проекта серьезные проблемы с векторизацией текстовых данных. Из-за этого очень сильно страдает выполнение работы в целом аплоть до невозможности выполнения некоторых операций, а именно:

* выбор алгоритма обучения модели;
* обучение модели;
* предсказание номинации для компании.

**1.3.1.2. Решение**

Полный пересмотр подхода к векторизации текстовых данных.

**1.3.2. Выбор модели**

Для выбора модели рассматривались следующие моменты:

\* перечислить модели с результатом

**1.3.2.1. Лучшая модель**

Модель и оценка

**2. API**

**2.1. Рассматриваются следующие типы API:**

* CLI
* GUI
* Web API

Каждый из этих типов подходит под задачу, но 2 особенно сильно выделяются:

* GUI
* Web-API

**2.1.1. GUI**

**Графический пользовательский интерфейс (graphical user interface, GUI**) — система средств для взаимодействия пользователя с электронными устройствами, основанная на представлении всех доступных пользователю системных объектов и функций в виде графических компонентов экрана.

Можно реализовать простой интерфейс с Событийно-ориентированной концепцией:

* Через каждый определенный промежуток обновлять данные;
* Доучивать модель;
* Обновлять списки номинантов.

При использовании данного метода, пользователю нужно:

* Скачать и установить приложение из общедоступного GitHub репозитория;
* Открыть приложение;
* Выбрать номинацию;
* Ознакомиться с номинантами

Недостатки данного подхода:

* Скачивание приложения только на ПК;
* Получение новых данных. Приложение сможет задействовать получение данных через сеть, что может приводить к существенному снижения скорости интернет соединения при получении обновлений.

**2.1.1.2. Web-API**

Web-API - это интерфейс программирования приложений для веб-сервера или веб-браузера. Как концепция веб-разработки, это может быть связано с клиентской частью веб-приложения.

Этот вариант будет приоритетней, потому что:

* Не нужно скачивать приложение;
* Относительно легкая реализация;
* Простота развертывания.

При использовании данного метода, пользователю нужно:

* Посетить веб-сайт;
* Выбрать интересующую категорию;
* Ознакомиться со списком номинантов.