САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

Факультет прикладной математики-процессов управления

Программа бакалавриата
“Большие данные и распределенная цифровая платформа”

ОТЧЕТ

по лабораторной работе №2

по дисциплине «Функциональное программирование»

на тему
«Распределенная обработка и анализ больших данных»
Вариант: 2

Студент гр. 23Б15-пу
Трофимов И.А.

Преподаватель
Киямов Ж. У.

Санкт-Петербург
2024 г.

Оглавление

Цель работы

Цель данной работы заключается в разработке эффективной программы для анализа больших объемов данных, получаемых из соцсетей, с использованием методов параллельных вычислений. Благодаря системе API мы можем получить доступ к данным Telegram, Vk и т.п. Необходимо написать программу, которая будет выкачивать данные из соцсетей и обрабатывать их.

Описание задачи (формализация задачи)

1. **Получение данных**

- Разработать модули для взаимодействия с API соцсетей (Telegram, VK и т.п.).
- Обеспечить поддержку многопоточного или асинхронного доступа к API для повышения скорости загрузки данных.
- Спроектировать структуру хранения данных, учитывая формат, предоставляемый API (JSON, XML и т.д.).

2. **Предобработка данных**

- Удаление шумовых данных:
- Удаление стоп-слов (например, "и", "или", "также").
- Удаление знаков пунктуации и лишних символов.
- Нормализация текста (приведение к единому регистру, лемматизация/стемминг слов).
- Фильтрация контента (например, исключение рекламных сообщений или ненужных категорий).

3. **Хранение данных**

- Выбор подходящей базы данных (SQL/NoSQL) для хранения структурированных и обработанных данных.
- Организация данных для эффективной индексации и последующего анализа.

4. **Анализ данных**

- Выделение ключевых слов, популярных тем, и трендов:
- Использование методов TF-IDF или моделей на основе Word2Vec/transformer.
- Анализ хэштегов и их взаимосвязей.
- Построение графов для определения трендов и связи пользователей/сообществ.

5. **Параллелизация и оптимизация**

- Реализация параллельной обработки данных:
- Разделение задач по сбору, предобработке и анализу данных между потоками/процессами.
- Использование библиотек для параллельных вычислений (например, `multiprocessing`, `joblib`, `asyncio`).
- Оптимизация использования оперативной памяти и ЦП.

6. **Визуализация результатов**

- Разработка модуля для создания графиков, диаграмм, и других визуальных отчетов.
- Построение интерактивных дашбордов с ключевыми метриками и трендами.

7. **Тестирование**

- Тестирование каждого модуля программы:
- Проверка корректности данных после их загрузки.
- Тестирование предобработки и анализа на небольших выборках.

8. **Документация**

- Описание структуры программы, включая схемы архитектуры.
- Инструкции по развертыванию и использованию программы.
- Комментарии к коду и примеры работы.

Теоретическая часть

1. Сбор данных

Программа осуществляет сбор данных из социальных сетей, таких как ВКонтакте и Telegram. Для работы с ВКонтакте используется библиотека vk_api, которая позволяет взаимодействовать с API платформы для получения постов из выбранных групп. В Telegram сбор данных выполняется с использованием асинхронной библиотеки Telethon, обеспечивающей быстрый и параллельный доступ к сообщениям в каналах. Основные этапы включают идентификацию групп и каналов по их именам, запрос данных через API, а также фильтрацию и обработку полученной информации.

2. Предобработка данных

Перед анализом текстовая информация проходит несколько этапов очистки и нормализации:

- **Удаление шумовых данных**: убираются специальные символы, знаки пунктуации, а также часто встречающиеся стоп-слова с использованием библиотеки nltk.
- **Нормализация текста**: текст приводится к нижнему регистру, а слова обрабатываются с учетом их лемм.
- **Фильтрация контента**: исключаются короткие слова и элементы, не несущие смысловой нагрузки.

Эти операции позволяют существенно уменьшить объем данных для анализа и выделить ключевые слова.

3. Анализ данных

Для анализа данных используется метод подсчета частотности слов:

- Программа формирует словарь, содержащий все слова из обработанных текстов.
- С помощью класса Counter из библиотеки collections рассчитывается количество вхождений каждого слова.
- Выводится топ-10 наиболее часто встречающихся слов для каждого источника, а также отдельно анализируются хэштеги в Telegram.

4. Хранение результатов

Результаты анализа сохраняются в текстовые файлы:

- Посты из ВКонтакте объединяются в один текстовый файл для удобства дальнейшей обработки.
- Результаты анализа (топ-слова и частотность) записываются в отдельные файлы с возможностью их просмотра через пользовательский интерфейс.

5. Параллелизация

Для увеличения производительности реализована параллельная обработка:

- ВКонтакте: используется модуль multiprocessing, позволяющий запускать несколько процессов для одновременного сбора и анализа данных из разных групп.
- Telegram: задействована асинхронная обработка данных с использованием asyncio и ThreadPoolExecutor. Это обеспечивает быстрое выполнение запросов к нескольким каналам одновременно.

6. Пользовательский интерфейс

Программа включает графический интерфейс, созданный с помощью библиотеки Tkinter. Пользователь может выбрать группы и каналы для парсинга, запустить процесс обработки данных, а также просмотреть результаты анализа. Интерфейс предоставляет удобные списки для выбора источников данных и кнопки для выполнения действий.

7. Особенности архитектуры

- **Модульность**: код разделен на функции, каждая из которых выполняет строго определенную задачу, что упрощает разработку, тестирование и расширение функциональности.
- **Асинхронность и многопоточность**: используется комбинация подходов для достижения максимальной эффективности при сборе и анализе больших объемов данных.
- **Поддержка расширяемости**: список групп и каналов может быть легко обновлен, а структура программы позволяет добавлять новые источники данных.

Основные шаги программы

1. **Запуск пользовательского интерфейса (GUI):**

    - Создается окно приложения с помощью библиотеки Tkinter.
    - Пользователь видит списки групп ВКонтакте и каналов Telegram, из которых можно выбрать источники для парсинга.
    - В интерфейсе предусмотрены кнопки для запуска парсинга и отображения результатов анализа.
2. **Выбор источников данных:**

    - Пользователь выделяет группы ВКонтакте и каналы Telegram в соответствующих списках.
    - При нажатии на кнопку парсинга выбранные группы и каналы передаются в обработку.
3. **Сбор данных из ВКонтакте:**

    - Для каждой выбранной группы вызывается функция, определяющая ее group_id через API ВКонтакте.
    - С помощью метода wall.get загружается до 100 последних постов для каждой группы.
    - Текст всех постов объединяется в один строковый объект.
4. **Сбор данных из Telegram:**

    - Асинхронно запрашиваются сообщения из выбранных каналов с помощью библиотеки Telethon.
    - Загружается до 100 сообщений для каждого канала.
    - Сообщения фильтруются для выделения текста и хэштегов.
5. **Предобработка текста:**

- Текст очищается от спецсимволов, знаков пунктуации и приводится к нижнему регистру.
- Из текста удаляются часто встречающиеся стоп-слова (например, предлоги и союзы) с помощью списка стоп-слов из библиотеки nltk.
- Очищенные слова проходят фильтрацию по длине (оставляются слова длиной больше двух символов).

6. **Анализ данных:**

   - Вычисляется частотность слов в тексте с помощью класса Counter.
   - Выводится топ-10 наиболее часто встречающихся слов.
   - В Telegram дополнительно выделяются и анализируются хэштеги.

7. **Сохранение результатов:**

   - Обработанные тексты из ВКонтакте записываются в файл all_vk_data.txt.
   - Результаты анализа для каждого источника записываются в текстовые файлы, где содержится топ-10 слов и их частотность.
   - Для Telegram создается отдельный файл с частотным анализом хэштегов.

8. **Отображение результатов:**

   - Пользователь может нажать кнопку для просмотра содержимого файлов с результатами.
   - Открывается новое окно с текстовым содержимым файла, что позволяет ознакомиться с результатами анализа.

9. **Асинхронная и параллельная обработка:**

    - Для ускорения обработки данных используются:
        - Модуль multiprocessing для работы с несколькими группами ВКонтакте одновременно.
        - Асинхронная обработка с помощью asyncio и Telethon для получения сообщений из нескольких Telegram-каналов.

10. **Завершение работы:**

    - После завершения всех операций выводится сообщение о завершении парсинга, уведомляющее пользователя о готовности результатов.

Описание программы

Таблица 1. main.py

Название функции	**Описание**	**Результат**
get_group_id(alias, token)	Определяет числовой group_id для группы ВКонтакте по ее короткому имени (alias) с использованием API ВКонтакте.	Возвращает числовой group_id группы или None в случае ошибки.
get_vk_posts(group_id, token, count=100)	Загружает последние count постов со стены группы ВКонтакте по group_id через API.	Список текстов постов группы.

collect_vk_data(source_name, group_id, token)	Объединяет текст постов ВКонтакте из указанной группы в один текст для дальнейшего анализа.	Возвращает строку с объединенным текстом всех постов или None, если данные не удалось получить.
preprocess_text(data)	Очищает текст: убирает спецсимволы, знаки пунктуации и стоп-слова. Преобразует текст в список очищенных слов.	Список очищенных слов, подходящих для анализа.
analyze_data(source_name, all_text, output_file)	Анализирует текстовые данные, вычисляет топ-10 наиболее частотных слов и записывает результаты в файл.	Сохраняет топ-10 слов с частотностью в файл.
vk_parser(selected_groups)	Организует сбор и анализ текстов постов из нескольких групп ВКонтакте. Использует многопоточность для ускорения работы.	Сохраняет объединенные данные и результаты анализа в файл all_vk_data.txt.
save_results_to_file(counter, file_name)	Сохраняет частотный анализ слов (или хэштегов) в текстовый файл в виде списка слов с их частотностью.	Создает файл с анализом слов или хэштегов.
telegram_parser(selected_channels)	Асинхронно собирает данные из сообщений нескольких каналов Telegram, выполняя частотный анализ текста и хэштегов.	Печатает топ-100 слов и хэштегов, сохраняет их в файлы counter_words.txt и других.

request_messages(channel_name)	Асинхронно загружает сообщения из указанного канала Telegram. Обрабатывает текст для выделения слов и хэштегов.	Возвращает два счетчика (Counter) — частотности слов и частотности хэштегов.
parse_data(selected_vk_groups, selected_telegram_channels)	Координирует сбор данных из выбранных источников ВКонтакте и Telegram, а также анализ и сохранение результатов.	Сообщает пользователю о завершении парсинга через окно сообщения.
show_file_content(file_name)	Отображает содержимое указанного текстового файла в отдельном окне с помощью графического интерфейса.	Открывает окно с текстовым содержимым файла или выводит сообщение об ошибке, если файл не найден.
create_gui()	Создает графический интерфейс для выбора источников (группы ВКонтакте и каналы Telegram), запуска парсинга и отображения результатов.	Запускает основное окно приложения с элементами управления.

Рекомендации пользователя

Для пользователя важно обеспечить доступ к выбранным источникам данных, настроить API-токены для ВКонтакте и Telegram, и убедиться, что компьютер соответствует техническим требованиям (Python 3.8+, необходимые библиотеки). Храните токены в безопасности, устанавливайте разумные лимиты на объем загружаемых данных, а результаты анализа изучайте через файлы с частотностью слов и хэштегов.

Рекомендации программиста

Программисту следует структурировать код, разделив его на модули, и снабдить комментариями. Важна обработка ошибок, безопасность токенов (использование .env файлов), оптимизация производительности через асинхронные запросы, логирование для отслеживания выполнения, и тестирование функций. Код должен быть документированным, а интерфейс интуитивно понятным. Также стоит предусмотреть масштабируемость для добавления новых источников данных и гибкость через использование конфигурационных файлов.

Исходный код программы

Jasur-Labs/fp/parser_tg_vk at main · hanglider/Jasur-Labs

Контрольный пример

При выполнении контрольного примера пользователь запускает программу и видит графический интерфейс, где выбирает группы ВКонтакте и Telegram-каналы для анализа. После выбора источников он нажимает кнопку "Запустить парсинг", инициируя сбор данных. Также можно выбрать объем данных, которые будут “выкачаны”. Программа отправляет запросы к API ВКонтакте для получения постов из указанных групп и к Telegram для сбора сообщений из выбранных каналов. Полученные данные пред обрабатываются: удаляются спецсимволы, стоп-слова, текст приводится к нижнему регистру, а слова сегментируются.

На основе очищенного текста программа вычисляет частотность слов и хэштегов, выделяя наиболее популярные. Эти данные записываются в файлы, которые пользователь может открыть, нажав соответствующую кнопку интерфейса. Для ВКонтакте создается файл с текстом постов и топ-10 слов, а для Telegram сохраняется файл с топ-100 словами и хэштегами. В завершение

пользователь видит сообщение о том, что данные успешно собраны и проанализированы. Рис (1)

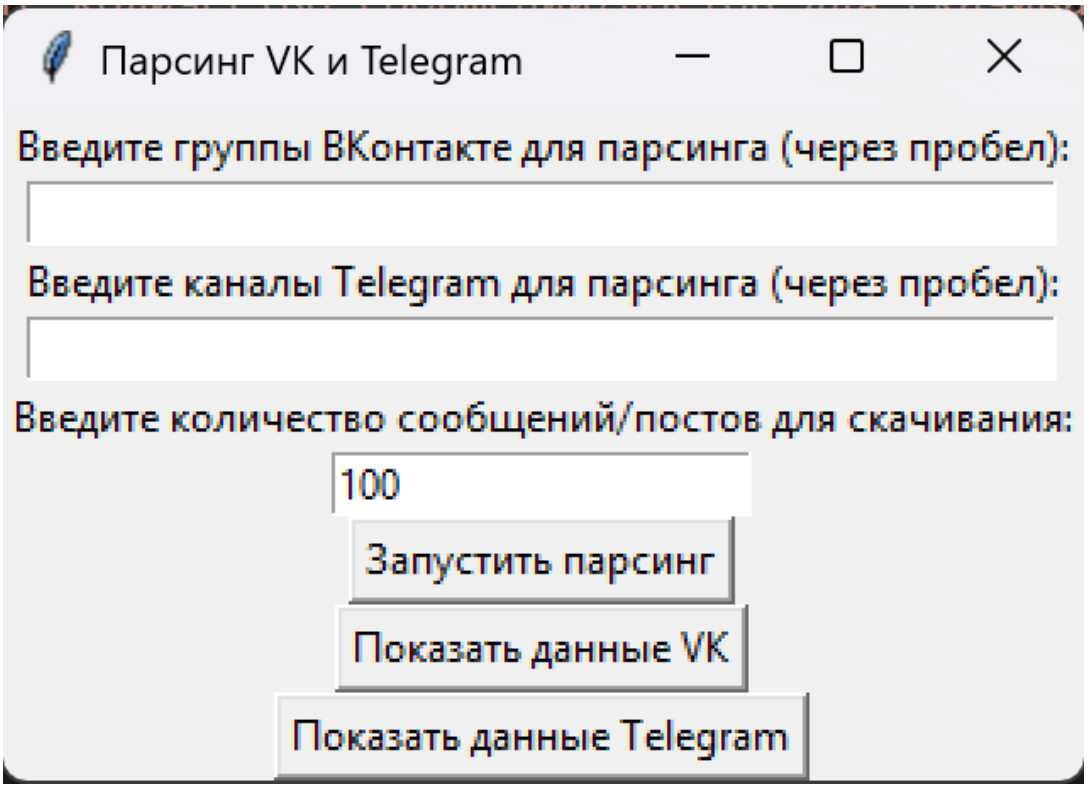

Рис 1. Графический интерфейс

Вывод

Программа успешно собирает, обрабатывает и анализирует текстовые данные из выбранных источников ВКонтакте и Telegram. Пользователь может легко получить доступ к ключевым результатам, включая частотный анализ слов и хэштегов, через понятный графический интерфейс. Это позволяет эффективно извлекать информацию о популярных темах и трендах из социальных сетей. Инструмент показал себя как удобное и универсальное решение для базового текстового анализа, что делает его полезным для исследователей, аналитиков и маркетологов.

Источники

- **OpenCV** https://docs.opencv.org/4.x/ *дата обращения: (09.11.2024)*

- **NumPy** *https://numpy.org/ дата обращения: (09.11.2024)*
- **tkinter** *https://docs.python.org/3/library/tkinter.html дата обращения: (09.11.2024)*
- **Pandas** *https://pandas.pydata.org/ дата обращения: (09.11.2024)*