МИНОБРНАУКИ РОССИИ
Федеральное государственное бюджетное образовательное учреждение
высшего образования
**«САРАТОВСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ
ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ИМЕНИ Н. Г. ЧЕРНЫШЕВСКОГО»**
Кафедра математической кибернетики и компьютерных наук

ПРИМЕНЕНИЕ ПОДХОДОВ BIGDATA ДЛЯ ПОСТРОЕНИЯ СЕТИ ЦИТИРОВАНИЯ СТАТЕЙ

АВТОРЕФЕРАТ МАГИСТЕРСКОЙ РАБОТЫ

Студентки 2 курса 273 группы
направления 02.04.03 — Математическое обеспечение и администрирование
информационных систем
факультета КНиИТ
Целикиной Милены Андреевны

Научный руководитель
к. ф.-м. н., доцент _____ С. В. Миронов

Заведующий кафедрой
к. ф.-м. н., доцент _____ С. В. Миронов

Саратов 2022

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Научные статьи являются важным объектом для оценки научной деятельности. Есть много методов для оценки и ранжирования статей. Для определения таких оценок необходимо иметь структурированную информацию о научных статьях.

Для анализа и оценки научных статей необходим датасет, который будет представлять из себя сеть (граф) цитирования статей. Датасеты часто необходимы для обучения нейронных сетей или хранения большого количества информации. Таким образом, нам необходим ориентированный граф, у которого вершинами являются статьи, а дуги ведут от статей к источникам, которые они цитирует. Специалисты по анализу данных и DataScience смогут провести необходимую обработку и аналитику такого набора данных и получить необходимые для их исследования и задач результаты.

Данная дипломная работа направлена на исследование возможности применения подходов BigData для построения сети цитирования статей и реализацию решения, способного применить такие подходы и методы.

Данная работа является актуальной, поскольку университету необходим инструмент, позволяющий строить подобные датасеты для задач анализа такого рода данных.

Целью дипломной работы является создание инструмента, позволяющего построить датасет статей на основе двух журналов в виде сети цитирования.

Цель настоящей дипломной работы порождает ряд задач, которые позволяют достигнуть поставленной цели:

- подобрать и изучить литературу, необходимую для решения данной задачи;
- проанализировать существующие подходы, инструменты и библиотеки для создания инструмента для построения сети цитирования;
- провести сравнительный анализ описанных выше инструментов и выбрать наиболее подходящие;
- разработать модули для извлечения необходимых данных с сайтов;
- создать модули, которые позволят предобработать данные;
- разработать инструмент для построения сети цитирования;
- разработать модуль для анализа и вывода данных в читаемом формате.

Объектом данного исследования является предметная область, т.е. теоре-

тические аспекты, необходимые для построения сети цитирования, и изучение возможности реализации решения с использованием подходов BigData.

Предмет исследования — инструменты и подходы BigData.

Дипломная работа состоит из 3 глав:

1. Задача построения сети цитирования.
2. Анализ и выбор инструментов для реализации и развёртывания решения.
3. Практическая часть.

1 Задача построения сети цитирования

1.1 Постановка задачи

Основной задачей данного исследования является построение сети цитирований статей. Исследование сетей цитирования — актуальная задача, поскольку научные статьи важны для анализа и оценки научной деятельности.

В настоящее время проводятся исследования в области анализа сетей цитирования. Также есть уже существующие решения для построения таких сетей цитирования. В качестве одного из решений можно представить инструмент «Springer SciGraph» , первый выпуск которого состоялся в 2017 году. Проблема решения подобной задачи поднимается в работах «Построение графовых моделей сети цитирования научных публикаций», «Задачи и методы автоматического построения графа цитирований по коллекции научных документов». Кроме того, университету в своих исследованиях необходим инструмент для построения сети цитирования.

Научная новизна исследования состоит в том, что в работе впервые изучена возможность использования подходов BigData для построения сети цитирования статей и разработано решение с применением этих подходов на практике.

1.1.1 Описание данных о статьях

В нашем случае вершины будут хранить данные о статьях. Данные о статье представляют собой следующую метаинформацию:

- название;
- авторы;
- дата публикации;
- название журнала и номер;
- DOI.

Изначально было выбрано DOI для однозначной идентификации статей, но его не всегда можно извлечь из полученных данных о статье, поэтому сейчас используется DOI, если он был получен с сайта, и название статьи в ином случае.

1.2 Анализ существующих решений

Одним из решений, которое было проанализировано, было решение В. А. Полежаева. В своей статье «Задачи и методы автоматического построения

графа цитирований по коллекции научных документов» он рассмотрел механизм идентификации статьи в библиографическом списке. Им был построен алгоритм выделения списка библиографии из различных статей и источников.

Данное решение хорошо подходит, если документы получены путём сканирования текста.

В нашем случае зачастую библиография уже получена, поэтому предварительная обработка ей не требуется. Также в данном решении не используется подход «BigData», а используется аналитический алгоритм, что может достаточно медленно работать на большом количестве данных.

Второе решение, которое было рассмотрено, является «Springer Nature SciGraph Explorer». Данный ресурс решает задачу в ограниченном объёме. К сожалению, построение графа ограничено лишь журналом «Springer Nature». Мы же хотим сделать наиболее универсальное решение.

Также рассмотрим решение из статьи «Построение графовых моделей сети цитирования научных публикаций». В работе используется обход в ширину веб-страниц с использованием фреймворков BeautifulSoup и Selenium. Такой подход имеет некоторые минусы. Во-первых, сбор информации в этом случае будет происходить достаточно медленно. С другой стороны на выходе у нас получается готовый граф, что является плюсом. Второй минус заключается в недостаточной гибкости такого подхода. Сайт может достаточно просто защитится от подобного сбора информации, всего лишь поменяв теги на странице. Таким образом, сбор информации превращается в гонку. В-третьих, данное решение ограничено одним ресурсом и с первого взгляда не является расширяемым, что также накладывает свои ограничения. Плюсом данного решения является интерактивность — данные собираются на ходу, что позволяет нам не хранить лишнюю информацию.

1.3 Анализ возможности сбора данных с сайтов

Первый ресурс, который был выбран, — «SpringerLink». Первым делом было опробовано решение на основе библиотеки `Scrapy`. Однако позднее на данном ресурсе была найдена возможность использования `API` данного сайта в исследовательских целях. «SpringerLink» имеет всю необходимую для нашей задачи информацию о научных статьях. Для использования открытого API необходимо запросить бесплатный API-ключ для исследовательских целей. Модераторы рассмотрят ваше предложение и пришлют такой ключ. Результат

запроса к API приходит в виде XML-файла с данными.

Второй ресурс, который был выбран — сайт «Public Library of Science (PLOS)». К сожалению, открытого `API` у данного сайта нет, поэтому был использован подход, схожий с подходом скрапинга. Отличие от классического скрапинга заключается в том, что в обычном скрапинге мы извлекаем необходимую информацию из html-кода. В нашем случае мы получаем данные из ответа на Ajax-запрос в виде json-файла.

1.4 Инструменты для реализации задачи

1.4.1 Инструменты для извлечения данных

Для этапа извлечения данных необходим такой язык, который способен обратиться к сайту и забрать данные при помощи специального запроса или путем извлечения их со страниц веб-ресурсов. Для таких задач идеально подходит язык Python. Он является одним из самых простых в освоении и популярных языков программирования. Кроме того, код Python ориентирован на высокую читаемость, а также на повышение производительности разработчика.

Для решения задачи подойдёт библиотека `requests`, которая позволяет общаться с сайтом посредством HTTP-запросов. Так мы можем сделать запрос GET к сайту, запрашивая все данные о статьях. При этом нам необходимо в URL сайта указать необходимый запрос и API-ключ. Нам вернётся результат в виде некоторой структуры: XML или JSON.

Кроме того, важную составляющую в приложении занимает разработка интерфейса для пользователя. Данная работа подразумевает разработку десктопного приложения. В языке Python есть немало библиотек для разработки подобных приложений.

1.4.2 Инструменты для обработки данных

Программа для построения сети цитирования предполагает гибкость и эффективность: она должна быстро обрабатывать тот объём данных, который придёт. В перспективе данные о статьях предполагается брать сразу из нескольких различных источников. Для выбора подходов к реализации прежде всего нам необходимо оценить объём обрабатываемых нами данных на всех этапах работы решения.

Научных статей сейчас очень много и их число только увеличивается. Обработать такой огромный поток данных (в том числе и неструктурированных) уже стало сложно или даже невозможно стандартными алгоритмами. Поэтому нам необходим мощный инструмент для работы с большими данными. Классическим инструментом для работы с BigData является экосистема Apache Hadoop.

Apache Hadoop — это каркас с открытым исходным кодом, предназначенный для создания и запуска распределенных приложений, обрабатывающих большие объемы данных.

Для обработки данных о статьях нам необходим такой инструмент, который сможет читать этот большой набор полуструктурированных или неструктурированных данных и эффективно его обрабатывать в пакетном режиме. В рамках Apache Hadoop есть несколько инструментов для таких задач:

- алгоритм MapReduce;
- фреймворк Apache Spark;
- инструмент Apache Flink.

Все 3 подхода имеют возможность работать с экосистемой Hadoop и обрабатывать большой объем данных, что нам и необходимо. Однако, мы знаем, что Spark пришёл на замену подходу MapReduce, поскольку MapReduce имеет ряд существенных недостатков перед Spark. Apache Flink и Spark во многом очень похожи: у них одно прикладное назначение и похожие особенности реализации кластерной обработки потоковых данных. Однако, Flink является менее популярным и используемым в продакшн решениях. Это в свою очередь приводит к тому, что у него есть много малоиспользуемых библиотек, которые находятся ещё в бета-режиме. Кроме того, довольно сложно его изучать, поскольку очень мало о нём простых и понятных статей, а документация не является развёрнутой и очевидной.

Поэтому решено использовать Apache Spark. Он является удобным и понятным инструментом для обработки больших данных, умеет работать с экосистемой Hadoop и довольно просто найти по нему информацию. В рамках задачи будем использовать высокоуровневое API Spark, представленное в виде DataFrame, поскольку с ним удобнее работать, а также оно является более эффективным за счёт оптимизатора Catalyst.

Для хранения данных был выбран инструмент HDFS. HDFS позволяет

хранить данные больших объёмов эффективно, а также позволяет получать быстрый доступ к ним и обеспечивает высокую отказоустойчивость. HDFS идеально подходит для задач, работающих с большим объёмом данных с помощью Spark.

Для реализации этапов обработки данных и построения сети цитирования статей необходимо выбрать наиболее подходящий для этих целей язык. Для данных этапов был выбран фреймворк Apache Spark.

Фреймворк Apache Spark написан на языке Scala, а поскольку Scala использует JVM (Java Virtual Machine), то язык Java также изначально поддерживался для Spark. Для языков Python и R поддержка Spark появилась гораздо позже. Поскольку Spark написан на Scala и имеет удобную интерактивную оболочку на этом языке, а также в связи с удобством разработки при использовании данного фреймворка перед другими языками, был выбран именно этот язык.

1.5 Инструменты для реализации UI

Для вывода данных и файла с графом в читаемом формате и проведения аналитики на основе него необходимо создать также десктопное приложение. Для этих целей выбран язык Python и фреймворк `PyQt5`.

Qt — это набор кроссплатформенных библиотек C++, которые реализуют высокоуровневые API для доступа ко многим аспектам современных настольных и мобильных систем. *PyQt5* — это полный набор привязок Python для Qt v5. Он позволяет использовать Python в качестве альтернативного языка разработки приложений для C++ на всех поддерживаемых платформах.

Фреймворк Qt позволяет разрабатывать интерфейс приложения как в ручном, так и в автоматическом режиме. При создании приложения разработчикам необходимо реализовывать взаимодействия между объектами в системе, при этом скрывая этот факт от самих объектов. Разработчики Qt создали концепцию сигналов и слотов. В Qt есть стандартные сигналы, такие как `clicked()`, `accepted()` или `valueChanged()`. Кроме того, можно реализовать и собственные сигналы. Для того, чтобы создать сигнал, необходимо использовать функцию `pyqtSignal()`.

1.6 Инструменты для развёртывания решения

Начнём рассмотрение инструментов для деплоя с Apache YARN. YARN (yet another resource negotiator) отвечает за управление ресурсами кластера и планирование заданий. Работать под управлением YARN могут как MapReduce-программы, так и любые другие распределённые приложения, поддерживающие соответствующие программные интерфейсы; YARN обеспечивает возможность параллельного выполнения нескольких различных задач в рамках кластера и их изоляцию.

Для решения задачи построения сети цитирования статей необходимо также реализовать инкрементальную загрузку новых статей. Такая загрузка предполагает наличие расписания для запуска приложения. Нам также необходимо создать пайплайн, который расположит задачи в нужном для корректной работы приложения порядке. В этом нам поможет Apache Airflow. *Apache Airflow* — это платформа для программного создания, планирования и мониторинга рабочих процессов. В Airflow рабочие процессы создаются в виде направленных ациклических графов (DAG) задач. В рамках нашей задачи будем использовать Airflow как планировщик для инкрементальной загрузки информации о новых статьях и построении на основе них и уже загруженных ранее данных нового графа.

Решения, построенные с использованием элементов Apache Hadoop, необходимо разворачивать на кластере, на котором все эти элементы должны быть развёрнуты и настроены. Для этого используем Docker. Он позволяет создать виртуализацию на уровне ОС, работать контейнерам как отдельным процессам ОС, ускорить разворачивание приложения, а также снизить расход памяти и сэкономить место на диске.

В Docker нам надо развернуть HDFS. Для этого возьмём Docker-образ в котором содержатся нужные нам компоненты. Основная проблема заключается в совместимости версий разных компонентов. Поскольку мы должны использовать версию Spark 3.2, необходимо выбрать совместимую версию HDFS. В нашем случае это версия 3.2.1. Для работы приложения нам требуется фреймворк Spark. Соберём образ Spark версии 3.2. Для запуска по расписанию нам требуется контейнер с airflow. У компоненты airflow должна быть своя копия Spark внутри, для запуска Spark-приложений. Она должна совпадать с той, которая находится в контейнере Spark.

2 Практические аспекты реализации

2.1 Структура решения

Полученное решение состоит из нескольких отдельных модулей, которые написаны с использованием разных языков и технологий. Все эти модули собраны воедино и запускаются в нужном порядке с помощью Apache Airflow.

На первом этапе параллельно запускаются 2 task: `extract_plos` и `extract_springer`, которые занимаются извлечением данных с источников в параллельном режиме. Для извлечения был использован язык программирования «Python». Проект содержит в себе три файла:

— `main.py` — здесь определяется с какого ресурса будет извлечение данных и временной интервал извлечения статей. На вход программе поступает аргумент, в результате обработки которого будет определено, из какого ресурса будут извлекаться данные. После будет создан экземпляр класса, нужного нам для извлечения данных с определённого ресурса;

— `ExtractorSpringer.py`;

— `ExtractorPLOS.py`.

Вторым этапом происходит предобработка данных, приведение в необходимый формат и складывание данных в нужное для последующей обработки место. Для этого созданы 2 task: `plos_to_parquet` и `springer_to_normal` для источников PLOS и Springer соответственно.

Следующим шагом необходимо все полученные с источников данные обработать и построить граф цитирования. За это отвечает task `increment`. Последним этапом необходимо удалить промежуточные данные. Это делается при помощи task `rm_raw`.

Для наглядного представления сети цитирований было создано небольшое десктопное приложение, которое позволяет пользователю просмотреть имеющиеся статьи, информацию о них, а также вывести информацию о статьях, которые являются источниками для выбранной. Кроме того, с помощью него можно произвести некоторую аналитику отношения цитирования статей.

2.2 Разработка способа инкрементальной выгрузки данных о статьях и построение итогового графа

Поскольку архитектура решения предполагает добавление данных к уже существующим, требовалось разработать способ инкрементальной загрузки.

Данные полученные ранее приведены к одной схеме, а значит теперь мы можем их обработать. Как было сказано ранее, для построения графа мы будем использовать фреймворк «Apache Spark».

Для этого создадим проект, который содержит в себе следующие файлы:

- Config.scala — содержит в себе информацию о пути к исходному файлу графа, о пути предобработанных данных и конечном пути графа в плоском виде;
- GraphFs.scala — находится класс, который содержит в себе схему данных графа, конфигурацию файловой системы Hadoop, и функцию создания DataFrame, который равен существующему графу;
- SparkAppCreatorGraph.scala — точка входа в программу.

Алгоритм построения сети цитирования предполагает два случая:

1. Граф ещё не создан. В этом случае граф строится только на основе полученных с источников статей.
2. Граф уже есть. Строим новый граф на основе существующего и полученных данных о статьях. Тогда алгоритм следующий:
   - среди новых статей и их источников оставляем только те, которых ещё нет в графе;
   - среди статей имеющихся в графе ищем те, информации о которых мало (это те статьи, которые были когда-то получены из библиографии других статей), и обогащаем их, если это возможно;
   - строим новый граф.

ЗАКЛЮЧЕНИЕ

В рамках данной дипломной работы была достигнута главная цель — создан инструмент, позволяющий построить датасет статей на основе двух журналов в виде сети цитирования.

Для достижения цели были решены следующие задачи:

- подобрана и изучена литература, необходимая для решения данной задачи;
- проанализированы существующие подходы, инструменты и библиотеки для создания инструмента для построения сети цитирования;
- проведён сравнительный анализ описанных выше инструментов и выбраны наиболее подходящие;
- разработаны модули для извлечения необходимых данных с сайтов;
- создан модуль, который позволяет предобработать данные;
- разработан инструмент для построения сети цитирования;
- доработан существующий модуль для анализа и вывода данных в читаемом формате.

Объектом данного исследования является предметная область, т.е. теоретические аспекты, необходимые для построения сети цитирования, и изучение возможности реализации решения с использованием подходов BigData.

Предмет исследования — инструменты и подходы BigData.

Таким образом, этот проект, действительно, является актуальным, поскольку университету необходим инструмент, который сможет предоставить датасет с информацией о цитировании статей для исследования научной деятельности. Разработанное приложение позволяет построить такой датасет.

В результате решения поставленной задачи были исследованы особенности построения сети цитирования статей и возможности реализации решения с использованием подходов BigData. Исследованы инструменты и подходы BigData.

Была написана и опубликована статья по материалам работы: «Применение подходов BigData для построения сети цитирования статей».