Санкт-Петербургский государственный университет

Математическое обеспечение и администрирование информационных систем

Группа 22.М04-мм

Модификация процессов сбора, хранения и управления данными для апробации алгоритмов анализа графов

Соболь Дарья Валерьевна

Отчёт по преддипломной практике

в форме «Решение»

Научный руководитель:
старший преподаватель, к.ф-м.н., Р.Ш. Азимов

Санкт-Петербург
2024

Оглавление

Введение

В современной информационной среде представление данных в виде помеченных графов играет ключевую роль в различных областях, таких как биоинформатика [13] или статический анализ кода [12, 15]. При работе с такими данными часто возникают задачи поиска путей, удовлетворяющих определенным ограничениям в виде формальных языков. Результаты обработки таких запросов обычно представляют собой набор отношений между вершинами графа. Один из естественных способов определить эти отношения — указать соответствующие пути, используя формальные грамматики над алфавитом меток ребер. Такие пути могут быть выражены с помощью контекстно-свободных грамматик. Наиболее популярны регулярные грамматики, но в последнее время активно разрабатываются алгоритмы для контекстно-свободных грамматик, так как они позволяют решать ряд задач, с которыми не справляются регулярные [16]. Возникает вопрос о необходимости разработки и реализации алгоритмов поиска путей с контекстно-свободными ограничениями.

Учитывая широкое применение контекстно-свободных грамматик в перечисленных выше областях, критически важной становится потребность в измерении производительности алгоритмов, реализующих эти запросы. Для демонстрации применимости алгоритма на практике требуется проведение экспериментального исследования на помеченных графах, соответствующих реальным данным. Однако поиск и подготовка таких графов весьма сложны и могут занять значительное время.

Модернизация хранилища данных для такого проекта контекстно-свободных грамматик к помеченным графам может принести несколько значительных преимуществ. Новая система хранения данных может быть оптимизирована для быстрого выполнения задач поиска путей в помеченных графах, что позволит ускорить и упростить экспериментальные исследования алгоритмов. Также, модернизированное хранилище данных может обеспечить более эффективную подготовку реальных помеченных графов для экспериментального исследования, что

упростит проведение тестов и экспериментов. В результате, модернизация хранилища данных может способствовать повышению точности и скорости анализа данных, необходимого для разработки и оптимизации алгоритмов контекстно-свободных грамматик к помеченным графам.

Один из таких инструментов — проект CFPQ_Data [5]. Этот проект предоставляет средства для работы с алгоритмами задачи поиска путей с ограничениями в виде формальных языков, что является важной задачей анализа помеченных графов. Однако, существующая инфраструктура для проекта CFPQ требует дальнейшего развития и модернизации инструментария.

Целью данной магистерской работы является модернизация и расширение функциональности проекта CFPQ_Data [5]. Это включает разработку и реализацию базы данных с использованием оптимизированной модели хранения данных для эффективного сбора и доступа к данным, а также проведения экспериментальных исследований. Новое хранилище данных планируется интегрировать в проект CFPQ_Data [5], что значительно улучшит взаимодействие с платформой.

Таким образом, модернизация проекта CFPQ_Data [5] будет способствовать развитию эффективных инструментов для анализа помеченных графов и улучшит возможности исследования в области ограниченного поиска путей на основе формальных языков.

1. Постановка задачи

Целью данной работы является модернизация проекта Context-Free Path Querying (CFPQ) Data [5], модернизация веб-версии, расширение его функциональности и интеграция базы данных для эффективного хранения и управления данными. Главной задачей проекта является создание современного и удобного инструмента для поддержки алгоритмов поиска путей, удовлетворяющих определенным ограничениям в виде формальных языков.

В рамках работы предусмотрены следующие этапы:

Исследование и анализ существующей веб-версии проекта CFPQ_Data [5] с целью выявления узких мест и потенциальных областей для улучшения. Это включает обзор конкурентных продуктов и существующих подходов к хранению и управлению данными в контексте подобных проектов.

Разработка и реализация распределенного хранилища с целью эффективного хранения информации о графах, запросах и результатах поиска путей, удовлетворяющих определенным ограничениям в виде формальных языков. Такой подход позволит обеспечить стабильное и масштабируемое хранение данных, необходимых для последующей обработки и анализа в рамках проекта.

Модификация веб-интерфейса для добавления функциональности управления данными, включая загрузку новых графов и грамматики, а также просмотр результатов запросов. Реализация механизмов аутентификации пользователей для обеспечения безопасного доступа к данным.

Изучение различных подходов и практик в области алгоритмов поиска путей, удовлетворяющих определенным ограничениями в виде формальных языков с целью анализа существующих решений и методов. Апробация и воспроизведение этих методов для создания графа, который может быть использован в качестве тестового случая. Формулирование соответствующей формальной грамматики для запросов, которая будет совместима с данным графом. После этого данные будут

интегрированы в инфраструктуру CFPQ_Data [5] для расширения её функциональности и возможностей.

В результате успешной реализации данной работы ожидается получение обновленного и функционального веб-интерфейса CFPQ_Data [5] с модернизированным хранилищем, улучшенной базой знаний, интеграцией нового графа и соответствующей грамматики, что позволит эффективно и гибко использовать алгоритмы поиска путей, удовлетворяющих определенным ограничениями в виде формальных языков.

2. Обзор

2.1. Хранение данных

В современном мире существует множество различных типов хранилищ данных, каждое из которых имеет свои уникальные особенности и преимущества. Реляционные базы данных, такие как PostgreSQL, MySQL и Oracle Database, являются стандартом для хранения структурированных данных с использованием SQL для запросов и управления данными. Они обеспечивают надежность, целостность данных и расширенные возможности аналитики.

NoSQL базы данных, такие как MongoDB, Cassandra и Redis, предоставляют гибкую модель данных без строгой схемы и могут эффективно обрабатывать неструктурированные данные. Они обеспечивают высокую производительность при работе с большими объемами данных и масштабируются горизонтально.

Графовые базы данных, включая Neo4j, ArangoDB и Amazon Neptune, специализируются на хранении и обработке данных в виде графов. Они предоставляют оптимизированные алгоритмы для работы с графами и выполнения специфических запросов, что делает их идеальными для анализа связей между данными.

Облачное хранение данных является одним из популярных на данный момент решений. Наиболее известными и доступными облачными хранилищами данных являются Amazon S3, Google Cloud Storage и Yandex Object Storage. Они обеспечивают высоко масштабируемое и доступное хранилище для файлов и данных. Эти хранилища гарантируют гибкость, высокую доступность и управление доступом к данным. Такие облачные хранилища данных как Yandex Object Storage (Yandex S3), представляют собой специализированные сервисы облачных вычислений, которые обеспечивают высоко масштабируемое и надежное хранение файлов и данных. Yandex S3 основан на технологии Amazon S3 (Simple Storage Service) и предлагает аналогичные возможности, оптимизированные для использования на российском рынке.

Взаимодействие с Yandex Object Storage осуществляется через HTTP-интерфейс, что делает его легко доступным для интеграции с различными приложениями и сервисами. Хранение данных в Yandex S3 организовано в виде объектов (файлов) и структур, обеспечивающих высокую степень отказоустойчивости и безопасности.

Ещё одним существующим решением для хранения данных, являются In-Memory базы данных. Например, Redis, Memcached и Apache Ignite, они хранят данные в оперативной памяти, обеспечивая высокую скорость доступа к данным и обработки запросов.

2.2. Обзор существующих проектов

Цель данного обзора состоит в том, чтобы провести обзор существующих аналогичных проектов, исследований или продуктов, связанных с темой настоящей работы. В данной работе будут рассматриваться только самые популярные аналоги, в которых собраны графы из различных областей, в том числе из тех, в которых возникают задачи поиска путей, удовлетворяющих определенным ограничениями в виде формальных языков. Подобный анализ позволит выявить особенности и преимущества существующих подходов, а также определить их недостатки и возможности для улучшения.

1. **Matrix Market & The SuiteSparse Matrix Collection (formerly the University of Florida Sparse Matrix Collection)**, доступный по адресу [7], представляет собой обширную коллекцию разреженных матриц. Эти данные используются для исследований в области численного анализа, оптимизации, и других областях науки и инженерии. Он предоставляет широкий набор данных, включая реальные примеры из различных приложений, такие как структурный анализ, термический анализ и электромагнетизм. SuiteSparse Matrix Collection ориентирован на математиков, инженеров и специалистов в области численного моделирования. Представленные разреженные матрицы широко применяются для эффективного представления и анализа данных в контексте раз-

личных задач. Коллекция SuiteSparse предоставляет исследователям и инженерам доступ к разнообразным данным, позволяя проводить тестирование алгоритмов и методов на реальных или сгенерированных данных. Также в веб-интерфейсе реализовано удобное и интуитивно понятное добавление новых данных в уже существующую коллекцию проекта.

2. **LDBC Social Network Benchmark (SNB)** [6] предоставляет наборы данных, которые моделируют социальные сети и предназначены для тестирования и оценки производительности графовых баз данных. Цель этого проекта – в создании стандартов и методологий для бенчмаркинга систем баз данных, что позволяет оценивать как скорость выполнения запросов к данным, так и другие аспекты производительности. Существующая коллекция наборов данных LDBC включает в себя сценарии реального времени. Например, обработка динамически изменяющихся графов (в социальных сетях с добавлением/удалением друзей, сообщений и т.д.). Этот ресурс может быть полезен для исследователей и разработчиков баз данных, стремящихся понять и улучшить производительность графовых систем. Однако, стоит отметить, что данный проект не является интуитивно понятным и удобным для пользователя. Затруднено взаимодействие с разработчиками, пополнение коллекции, а также просмотр сводной статистики по имеющимся данным.

3. **CFPQ_Data repository**, доступный по адресу [5], фокусируется на предоставлении данных для запросов на графах с ограничениями, выражаемыми контекстно-свободными грамматиками. Этот проект находится в стадии активного развития в области обработки грамматик к графам и базам данных, где особый интерес представляют грамматики, требующие анализа сложных шаблонов и связей между узлами графа. CFPQ_Data предоставляет данные, основанные на реальных и синтетических графах, что помогает исследователям разрабатывать и тестировать алгорит-

мы обработки грамматик с ограничениями в виде формальных языков.

Каждый из этих проектов имеет свою уникальную цель и направленность, которая определяет их применимость в различных областях научных исследований и разработок. Matrix Market и SuiteSparse предоставляют данные для численного анализа и оптимизации, LDBC SNB моделирует социальные сети для тестирования графовых баз данных, а CFPQ_Data специализируется на запросах на графах с использованием контекстно-свободных грамматик. Выбор этих аналогов в обзоре обусловлен их значимостью и репутацией в соответствующих областях исследований запросов на графах и анализа данных.

3. Реализация

3.1. Модификация веб-интерфейса

Для начала, цель работы определена как создание системы хранения для результатов обработки различных графовых структур с последующей возможностью фильтрации и представления данных пользователям через интерфейс веб-сайта. В качестве основы для системы хранения были рассмотрены различные базы данных, включая PostgreSQL [9] и ClickHouse [2], с целью определения наиболее подходящего решения с точки зрения производительности, удобства использования и масштабируемости [11].

Проведенное сравнительное исследование между PostgreSQL и ClickHouse позволило выявить ряд преимуществ PostgreSQL, в частности, его высокую надежность, поддержку транзакций и богатый функционал для работы с сложными запросами, что обеспечило выбор в пользу PostgreSQL.

Для каждой пары (граф, грамматика), где граф может быть подвергнут грамматике, необходимо сохранить результаты в удобной форме. Например, для задачи достижимости нужно хранить множество вершин (список чисел).

Поэтому для хранения была предложена таблица. Она позволяет выполнять запросы на фильтрацию данных. Например, фильтровать данные по количеству ребер, классу грамматики, типу грамматики и так далее.

Предположим, что есть графы g1, g2, g3, и грамматики q1, q2. Грамматика q1 применима ко всем графам, в то время как грамматика q2 применима только к g2 и g3. Для каждой такой комбинации (граф, грамматика) необходимо хранить результаты, чтобы можно было проводить быстрый поиск и предоставить результаты для всех отфильтрованных комбинаций. Пример таких комбинаций может выглядеть так: (g1, q1), (g2, q1), (g3, q1), (g2, q2), (g3, q2).

Для эффективного моделирования сложных отношений между графиками и грамматиками была создана база данных, содержащая таб-

лицы «graphs», «grammars» и «graphs_grammars». В данной базе данных применяется подход связи «многие-ко-многим», который позволяет устанавливать отношения между конкретными графами и грамматиками.

Связь «многие-ко-многим» представляет собой эффективный способ организации данных в базе данных, позволяя устанавливать множество отношений между элементами из различных таблиц. Такой тип связи таблиц предполагает, что один элемент из одной таблицы может быть связан с несколькими элементами из другой таблицы и наоборот. Это обеспечивает гибкость и расширяемость при моделировании сложных отношений.

Использование связи «многие-ко-многим» при создании базы данных обеспечивает эффективное описание сложных взаимосвязей между графами и грамматиками. Например, один граф может быть связан с несколькими грамматиками, а одна грамматика может соответствовать нескольким графам. Такой подход обеспечивает эффективное хранение и извлечение данных, обеспечивая гибкость при определении отношений между объектами, где каждый объект может быть связан с несколькими другими объектами без жестких ограничений.

Хранение самих данных было решено оставить в Yandex S3, поскольку продукт предоставляет масштабируемое решение для хранения больших объемов данных, что важно для проектов, требующих хранение и обработку графов и грамматик к ним. Гибкость и масштабируемость Yandex Cloud S3 позволяют эффективно управлять изменяющимся объемом данных и обеспечивают простоту интеграции с другими сервисами обработки данных.

Таким образом, комбинация PostgreSQL и Yandex S3 предоставляет надежное и масштабируемое решение для управления данными, включая возможность хранения и работы с большими объектами, что делает его подходящим выбором для целей данного проекта.

Особое внимание в работе уделено разработке механизмов для эффективного хранения и обработки результатов граматик и графов. Как было упомянуто, при обработке графовых структур возникает необхо-

димость в хранении результатов обработки в виде, поддерживающем быстрый доступ и эффективное использование данных. В рамках решения этой задачи был использован специализированный формат хранения, оптимизированный для хранения результатов, таких как множества достижимых вершин.

Дополнительно, в рамках данной работы была проведена модернизация веб-интерфейса, на котором результаты обработки графов представляются пользователям. Во-первых, была изменена структура проекта с целью повышения его модульности и читаемости. Это позволило лучше организовать код и разделить функциональные блоки на отдельные компоненты. Такой подход важен для обеспечения масштабируемости и поддерживаемости проекта в дальнейшем.

Кроме того, была реализована серверная часть для взаимодействия с базой данных (БД). Данный этап включает в себя создание API (интерфейса программирования приложений). Этот интерфейс позволяет взаимодействовать между клиентской частью (фронтендом) и базой данных. С его помощью можно получать доступ к данным и изменять их, отправляя соответствующие запросы.

Кроме того, были созданы маршруты для базы данных, которые указывают точки входа для обработки запросов к данным. Благодаря этому клиентская часть может отправлять HTTP-запросы на сервер с определёнными параметрами и получать соответствующие данные из базы данных.

Чтобы улучшить пользовательский опыт (UI/UX), были проведены следующие работы:

- Создание таблицы в веб-интерфейсе для удобного отображения данных из базы данных. Это включало разработку пользовательского интерфейса (UI), который позволяет просматривать данные в виде таблицы с возможностью сортировки и фильтрации;

- Внедрение функций фильтрации и сортировки для повышения удобства работы пользователей с данными. Это даёт возможность

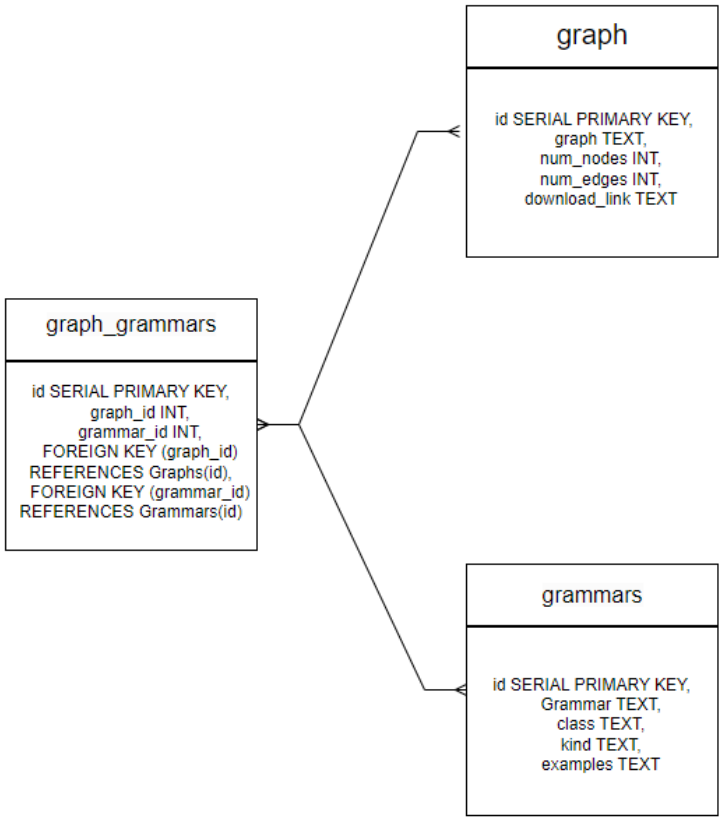

Рис. 1: Схема базы данных.

пользователям быстро находить нужную информацию и настраивать её отображение в таблице;

- Добавление возможности загружать новые материалы в базу данных через веб-интерфейс. Это позволяет пользователям динамически увеличивать содержимое базы данных, улучшая функциональность и полезность системы.

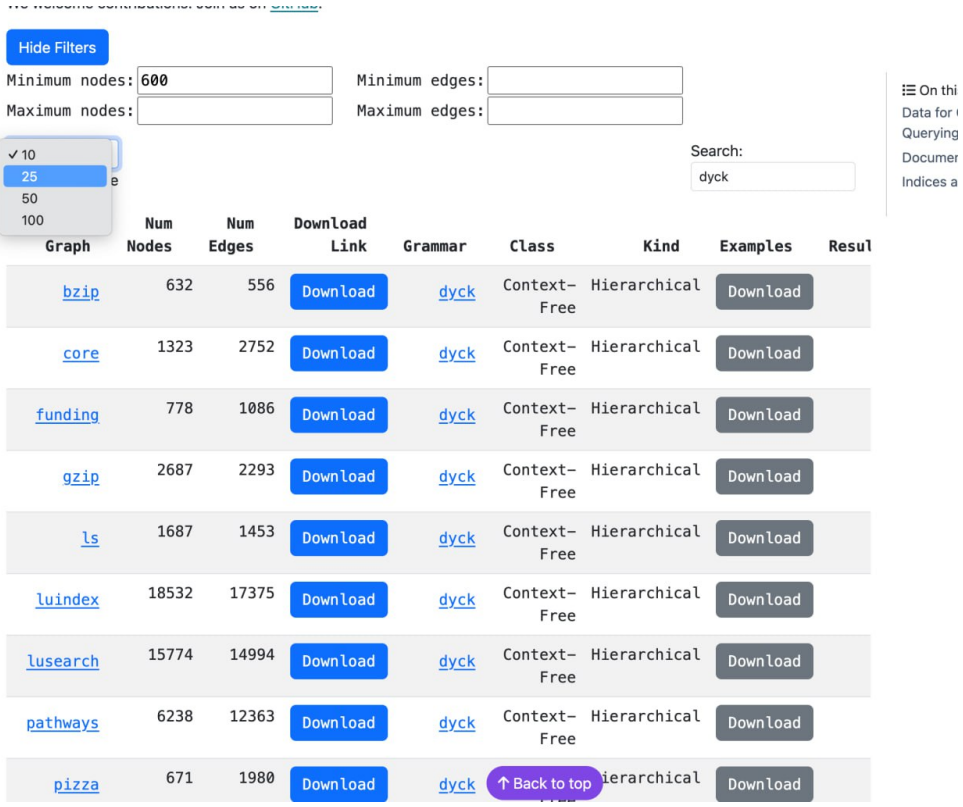

Рис. 2: Реализация фильтрации.

Важность UI/UX состоит в создании удобного и интуитивно понятного интерфейса для пользователей. Это позволяет повысить удовлетворённость пользователей и эффективность работы с программным продуктом. Для достижения этой цели можно использовать современные методы дизайна, обеспечить адаптивность интерфейса для различных устройств и оптимизировать процесс взаимодействия пользователя с приложением.

В результате проведенных исследований и разработок была создана эффективная и удобная в использовании система для хранения, обра-

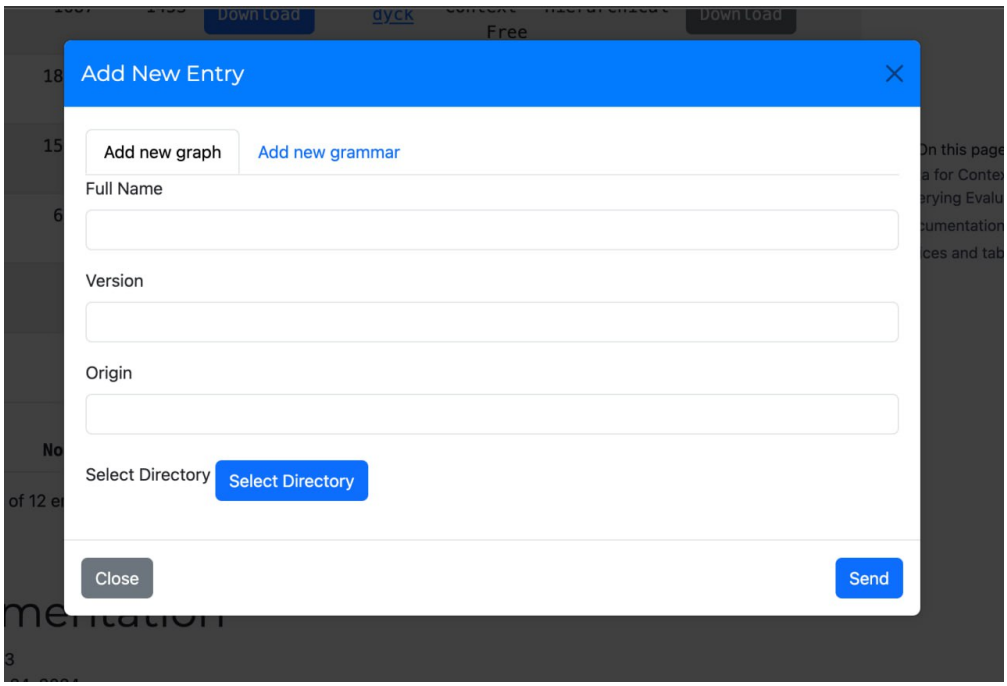

Рис. 3: Дополнительный функционал, упрощающий взаимодействие пользователя с системой.

ботки и визуализации результатов обработки графовых структур, которая позволяет пользователям в реальном времени получать и анализировать данные о работе алгоритмов поиска путей, удовлетворяющих определенным ограничениями в виде формальных языков. Таким образом, выполненная работа существенно улучшает возможности по работе с проектом CFPQ_Data и дает возможность использовать этот мощный инструмент для исследований в данной области.

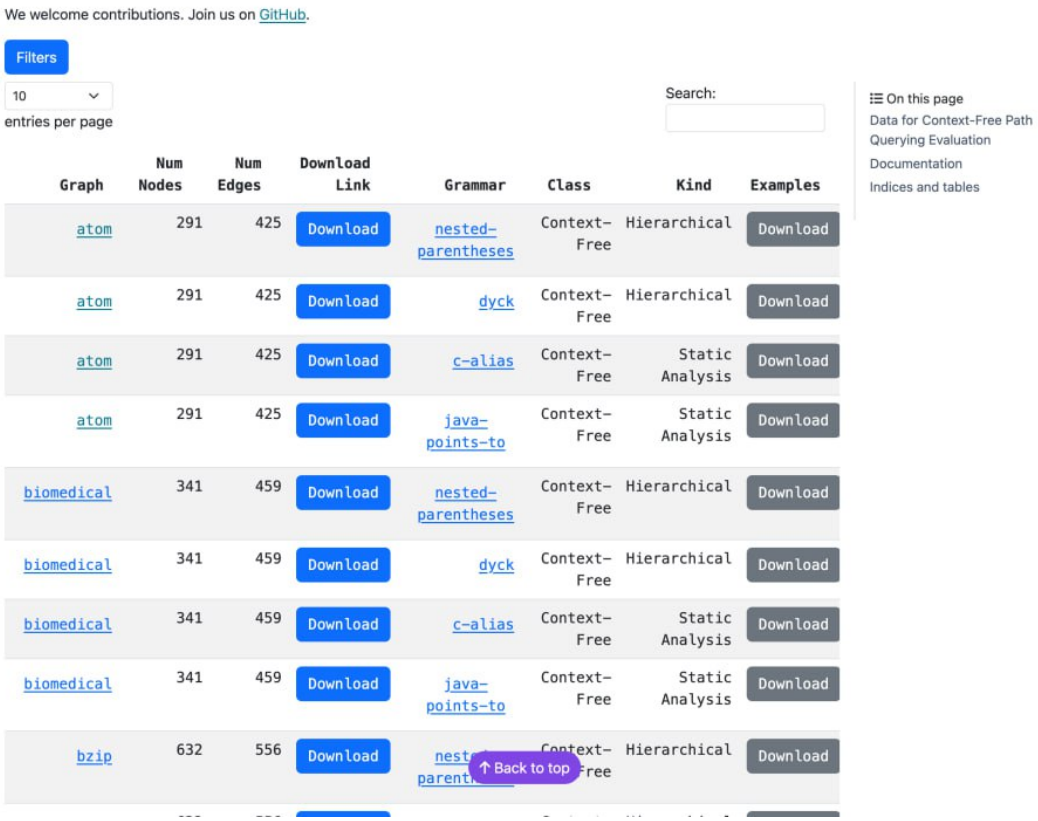

Рис. 4: Модифицированный веб-интерфейс проекта CFPQ-Data.

4. Эксперимент

В данном разделе описывается изучение области data provenance, эксперимент по процессу сбора данных (графов и грамматик), а также использование добавленного функционала для расширения набора данных CFPQ_Data.

4.1. Изучение области

Рассмотрим методы, подходы и результаты, описанные в работах из области data provenance. Основная цель этого подраздела - извлечение полезной информации и определения ключевых аспектов, влияющих на решение задачи сбора и анализа данных. Данные, восстановленные по описанным далее источникам, важны для развития проекта, модернизируемого в рамках данного проекта, т.к. они обеспечивают основу для дальнейшего анализа и построения собственных исследовательских подходов.

Provenance — это хронология владения, хранения или местоположения исторического объекта. Объекты включают в себя произведения искусства, публикации и артефакты в научных коллекциях [1]. С примером набора данных из области data provenance можно ознакомиться по ссылке [14].

1. В работе «Understanding Data Science Lifecycle Provenance via Graph Segmentation and Summarization» [8] были использованы разнообразные данные, включая синтетические графы provenance с различными характеристиками проекта. Эти графы были сгенерированы с использованием современных платформ Data Science [10], которые способны собирать информацию о provenance и контексте данных. Эти платформы также обеспечивают возможность обработки изменений одного и того же файла через различные версии.

   Кроме того, в статье рассматривались реальные данные из различных приложений, включая идентификацию владельца документов provenance, оценку качества краудсорсных данных. Подобный подход позволил авторам работы получить богатый набор данных для исследования и анализа, покрывающий различные аспекты работы с provenance в рамках жизненного цикла Data Science.

   В статье также дана оценка предложенным методам и продемонстрирована их эффективность. Данное исследование может быть полезно в управлении сложными совместными процессами анализа данных, а также в упрощении самого процесса анализа данных. Кроме того, исследование поможет в извлечении полезной информации и восстановить данные для проекта CFPQ.

2. Статья «ProvGen: Generating Synthetic PROV Graphs with Predictable Structure» [4] описывает инструмент ProvGen [3] для генерации синтетических графов provenance с предсказуемой структурой. Основная цель этого инструмента - создание наборов синтетических графов provenance, которые могут быть использованы для

тестирования алгоритмов поиска путей, удовлетворяющих определенным ограничениями в виде формальных языков.

Авторы статьи подробно описывают подход к созданию синтетических графов provenance с помощью инструмента ProvGen. Этот инструмент обладает гибкостью настройки параметров генерации графов, таких как размер, плотность и структура, что позволяет создавать графы, соответствующие различным сценариям исследований.

ProvGen генерирует графы с предсказуемой структурой. Это даёт исследователям контроль над проведением экспериментов и сравнением результатов с различными наборами данных. Благодаря этому, исследователи могут создавать синтетические графы, которые отвечают определённым исследовательским требованиям. Такие графы могут быть использованы для оценки эффективности алгоритмов поиска путей, основанных на графах provenance.

4.2. Сбор данных

Для сбора данных были проведены эксперименты, опираясь на статьи, которые исследованы в процессе ознакомления с областью.

Происхождение данных (Data Provenance): Как уже упоминалось, происхождение данных (provenance) в информационных системах означает отслеживание истории данных, включая их источники, преобразования и обработку, которая привела к конечным результатам. Происхождения может рассматривать историю данных в контексте систем обработки данных, таких как реляционные базы данных (RDBMS) и распределенные вычислительные платформы (например, Apache Spark). В статье [8] упоминается, что системы потока данных (dataflow systems) играют важную роль в отслеживании происхождения данных. Они позволяют понять, как данные были созданы, преобразованы и использованы в рамках вычислительных экспериментов.

В исследовательских проектах представляется в виде ориентированного ациклического графа (DAG):

$\mathcal{G}(\mathbb{V}, \mathbb{E}, \lambda_v, \lambda_e, \sigma, w)$,

где вершины имеют три типа $\mathbb{V} = \varepsilon \cup \mathcal{A} \cup \mathcal{U}$, а ребра имеют пять типов $\mathbb{E} = U \cup G \cup S \cup A \cup D$.

$\lambda_v : \mathbb{V} \mapsto \{\mathcal{E}, \mathcal{A}, \mathcal{U}\}$ и $\lambda_e : \mathbb{E} \mapsto \{U, G, S, A, D\}$ – суммарные функции, связывающие каждую вершину и каждое ребро с его типом.

В статье [8] исследования происхождения (provenance) категоризуется на два основных типа: происхождение данных (data provenance) и происхождение рабочего процесса (workflow provenance).

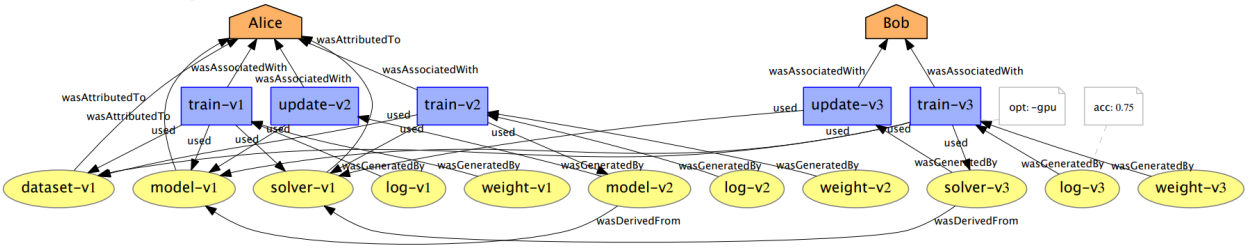

Рис. 5: Пример графа происхождения.

Происхождение рабочего процесса (Workflow Provenance): Workflow provenance изучает историю выполнения сложных рабочих процессов, таких как научные эксперименты, бизнес-процессы и задачи кибербезопасности. Она фокусируется на высокоуровневых концептуальных компонентах, их взаимодействиях и влиянии на конечные результаты. В статье отмечается, что workflow provenance имеет важное значение для понимания сложных взаимодействий в рабочих процессах и обеспечения их воспроизводимости и управления.

Статья подчеркивает различия между системами происхождения данных и системами происхождения рабочего процесса. В частности, workflow provenance исследует сложные взаимодействия между компонентами рабочего процесса и учитывает изменчивость в выполнении задач, что требует разработки эффективных методов для отслеживания и анализа происхождения результатов.

Статья также обсуждает актуальные тенденции и направления исследований в области происхождения, такие как разработка методов запросов и визуализации для анализа эволюционирующих происхождений скриптов (script provenance), что имеет особое значение в контексте

безопасности и воспроизводимости данных и результатов выполнения скриптовых сценариев.

На основе этой статьи создадим данные (графы, грамматики) для проекта CFPQ_Data, для этого составим техническое решение.

Техническое решение для генерации синтетического графа provenance (PROV) основывается на определенных параметрах и процессах моделирования, которые имитируют активности и связи между сущностями в рамках исследуемой системы provenance. Данное техническое решение является общим выявленным паттерном для сбора грамматик и графов. В результате нашего эксперимента планируется получить конкретный список графов, их характеристики, а также конкретные грамматики для этих графов.

1. Инициализация параметров:

   N: Желаемое количество вершин (entities) в графе.

   sw, λi, λo, se: Параметры для моделирования распределений. sw используется для выбора агентов, λi и λo - для генерации количества входных и выходных сущностей, а se контролирует вероятность выбора сущности в зависимости от ее ранга.

2. Генерация агентов (agents):

   Вычисляем количество агентов U как округленное значение **log(N)**.

3. Генерация сущностей (entities) и деятельностей (activities):

   Создаем пустой список entities для хранения сгенерированных сущностей.

   Для каждой деятельности генерируем num_activities деятельностей:

   Случайным образом выбираем агента из списка agents с использованием Zipf распределения (sw).

   Генерируем случайное количество входных (m) и выходных (n) сущностей с помощью распределения Пуассона (λi и λo соответственно).

Добавляем деятельность в список activities, указывая агента, список входных и выходных сущностей.

Добавляем сгенерированные выходные сущности в список entities.

4. Построение графа prov_graph:

   Создаем пустой словарь prov_graph для представления графа.

   Для каждой деятельности в списке activities и для каждой входной сущности в деятельности:

   Добавляем связь от входной сущности к каждой выходной сущности этой деятельности в prov_graph.

5. Возврат сгенерированного графа prov_graph:

   Возвращаем prov_graph как синтетически сгенерированный граф, содержащий связи между сущностями и активностями.

PgSeg запросы на сгенерированном графе:

1. Выбор PgSeg запросов (Vsrc, Vdst):

   Из сгенерированного графа выбираем пары вершин (Vsrc, Vdst) для выполнения PgSeg запросов.

   Обычно Vsrc выбираются как первые две сущности (вершины), а Vdst - как последние две сущности, так как они обычно связаны между собой.

2. Выполнение PgSeg запроса:

   Для каждой выбранной пары (Vsrc, Vdst):

   Используем алгоритм PgSeg для определения существования пути между Vsrc и Vdst в графе.

   Это включает различные стратегии поиска пути, включая эффективное обходное прохождение графа для определения связей между сущностями.

Это техническое решение позволяет создавать и анализировать синтетические графы, моделируя активности и их связи в контексте provenance данных. Применение PgSeg запросов позволяет эффективно исследовать связи и зависимости между сущностями в сгенерированном графе для аналитических и исследовательских целей.

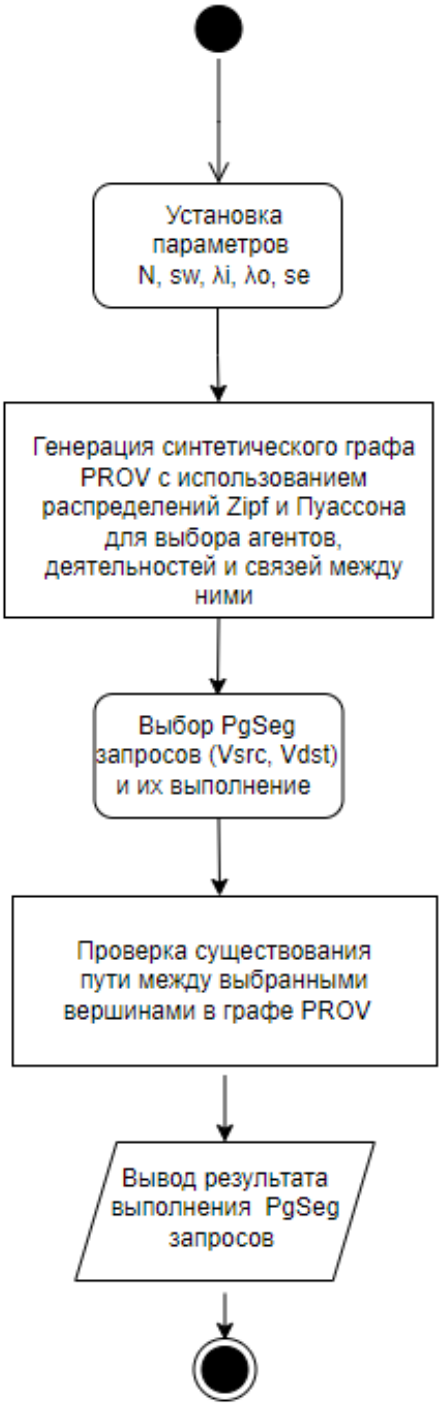

Рис. 6: Общая последовательность шагов для генерации синтетического графа.

Заключение

Данная работа направлена на модернизацию и расширение функциональности проекта Context-Free Path Querying (CFPQ) Data [5], с улучшением веб-интерфейса, расширением возможностей и внедрени-

ем базы данных для эффективного хранения и управления данными. Основной целью проекта является создание современного и удобного инструмента для поддержки алгоритмов поиска путей, удовлетворяющих определенным ограничениями в виде формальных языков.

В рамках данного проекта были реализованы следующие этапы:

- Исследование и анализ существующей веб-интерфейса CFPQ_Data:

  Проведено обширное исследование существующей версии CFPQ_Data, выявлены узкие места и потенциальные области для улучшения. Был проведен обзор конкурентных продуктов, их подходов к хранению и управлению данными. Данный анализ помог определить лучшие практики для дальнейшего развития проекта.

- Разработка и реализация распределенного хранилища:

  Разработано и реализовано распределенное хранилище для эффективного хранения информации о графах, грамматиках и результатах выполнения алгоритмов. Этот подход обеспечивает стабильное и масштабируемое хранение данных, необходимых для дальнейшей обработки и анализа.

- Модификация веб-интерфейса:

  В веб-интерфейс проекта была добавлена функциональность управления данными через веб-интерфейс, включая загрузку новых графов и грамматик. Также были реализованы механизм аутентификации пользователей для обеспечения безопасного доступа к данным.

- Изучение различных методов поиска путей, удовлетворяющих определенным ограничениями в виде формальных языков:

  Изучены различные подходы и практики в области алгоритмов поиска путей, удовлетворяющих определенным ограничениями в виде формальных языков. Проведено изучение области data provenance и эксперимент по сбору данных, а также использование добавленного ранее функционала для расширения набора данных CFPQ_Data.

Таким образом, получен обновленный веб-интерфейс CFPQ_Data с модернизированным хранилищем данных, улучшенной базой знаний и интеграцией нового графа и соответствующей грамматики. Это позволит эффективно и гибко апробировать алгоритмы поиска путей, удовлетворяющих определенным ограничениями в виде формальных языков в контексте данного проекта.

Список литературы

[1] Bio2RDF Dataset Provenance. — URL: https://github.com/bio2rdf/bio2rdf-scripts/wiki/Bio2RDF-Dataset-Provenance#bio2rdf-provenance-model,urldate={\DTMdate{2024-05-11}},language={english}.

[2] ClickHouse Docs. — https://clickhouse.com/docs/ru. — Accessed: 2023-10-10.

[3] Firth Hugo. Prov-gen. — URL: https://github.com/hugofirth/prov-gen,urldate={\DTMdate{2024-05-11}},language={english}.

[4] Firth Hugo, Missier Paolo. ProvGen: Generating Synthetic PROV Graphs with Predictable Structure. — Vol. 8628. — 2014. — 06.

[5] Formal Language Constrained Path Querying Data Repository. — https://formallanguageconstrainedpathquerying.github.io/CFPQ_Data/. — Accessed: 2024-04-05.

[6] LDBC Dataset: SURF. — https://ldbcouncil.org/data-sets-surf-repository/. — Accessed: 2023-10-10.

[7] Matrix Market The SuiteSparse Matrix Collection (formerly the University of Florida Sparse Matrix Collection). — URL: http://sparse.tamu.edu/ (дата обращения: 16 сентября 2023 г.).

[8] Miao Hui, Deshpande Amol. Understanding Data Science Lifecycle Provenance via Graph Segmentation and Summarization. — 2018. — 1810.04599.

[9] PostgresPro. — URL: https://postgrespro.ru/docs (дата обращения: 16 января 2024 г.).

[10] ProvToolbox. — URL: https://github.com/lucmoreau/ProvToolbox,urldate={\DTMdate{2024-05-11}},language={english}.

[11] R. Mattison. Data Warehousing and Data Mining for Telecommunications. — 1997.

[12] Rehof Jakob, Fähndrich Manuel. Type-base flow analysis: from polymorphic subtyping to CFL-reachability // SIGPLAN Not. — 2001. — jan. — Vol. 36, no. 3. — P. 54–66. — URL: https://doi.org/10.1145/373243.360208.

[13] Sevon Petteri, Eronen Lauri. Subgraph Queries by Context-free Grammars // Journal of Integrative Bioinformatics. — 2008. — Vol. 5, no. 2. — P. 157–172. — URL: https://doi.org/10.1515/jib-2008-100.

[14] Wikipedia-PROV. — URL: https://github.com/provbench/Wikipedia-PROV,urldate={\DTMdate{2024-05-11}},language={english}.

[15] Zheng Xin, Rugina Radu. Demand-driven alias analysis for C // Proceedings of the 35th Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages. — POPL '08. — New York, NY, USA : Association for Computing Machinery, 2008. — P. 197–208. — URL: https://doi.org/10.1145/1328438.1328464.

[16] Азимов Р.Ш. Григорьев С.В. АЛГОРИТМ ПОИСКА ВСЕХ ПУТЕЙ В ГРАФЕ С ЗАДАННЫМИ КОНТЕКСТНО-СВОБОДНЫМИ ОГРАНИЧЕНИЯМИ С ИСПОЛЬЗОВАНИЕМ МАТРИЦ С МНОЖЕСТВАМИ ПРОМЕЖУТОЧНЫХ ВЕРШИН // Научно-технический вестник информационных технологий, механики и оптики. 2021. №4. — URL: https://cyberleninka.ru/article/n/algoritm-poiska-vseh-putey-v-grafe-s-zadannymi-kontekstno-svobo urldate={\DTMdate{2024-05-11}},language={russian}.