Санкт-Петербургский государственный университет

Кафедра системного программирования
Программная инженерия

Ахметьянов Азат Ришатович

Создание набора данных для экспериментального анализа алгоритмов вычисления контекстно-свободных запросов

Курсовая работа

Научный руководитель:
к.ф.-м.н., доц. Григорьев С. В.

Санкт-Петербург
2020

Оглавление

Введение

Графовые структуры находят применение в биоинформатике, статическом анализе кода и многих других областях, все более широко используемыми становятся графовые базы данных. При работе с графами возникает задача навигации и поиска путей. Одно из решений проблемы навигации по графу заключается в использовании контекстно-свободных (КС) языков [23]. Путь в графе можно рассматривать как слово в языке — тогда запрос пути является спецификацией языка, то есть КС грамматикой. Задача алгоритма вычисления КС запросов путей заключается в том, чтобы найти такие пути в графе, что соответствующие им слова лежат в заданном языке.

Ввиду применения КС запросов в перечисленных выше практических областях, производительность алгоритмов их вычисления имеет критическое значение. Чтобы показать преимущества нового алгоритма вычисления запросов, необходимо провести экспериментальный анализ на некоторых данных, моделирующих реальные сценарии его применения. Такой анализ также позволяет исследователям сравнить производительность своего решения с представленными ранее алгоритмами.

Поиск, подготовка данных для проведения эксперимента отнимают продолжительное время. Для решения этих проблем во многих областях исследований используются стандартные наборы данных, упрощающие экспериментальный анализ алгоритмов. В области КС запросов на данный момент не существует такого набора, о его создании и пойдет речь в данной работе.

1. Постановка задачи

Цель работы — создание репозитория, содержащего данные для экспериментального анализа алгоритмов вычисления КС запросов в максимально удобном для исследователей виде. Данные включают в себя КС запросы и графы из различных областей.

Для достижения цели поставлены следующие задачи:

- Унификация представления уже добавленных данных.
  - Выбор единого формата всех графов.
  - Разработка человекочитаемого формата запросов.
- Добавление возможности получать из запросов грамматики в Нормальной Форме Хомского [2].
- Создание сайта проекта, индексируемого в Dataset Search [11] — поиске наборов данных для исследований от Google.
- Создание детальной документации с описанием представленных графов и грамматик, контрольными значениями вычисления запросов, а также ссылками на исследования, использующие эти данные для экспериментального анализа.
- Добавление возможности загружать набор данных в следующие графовые БД:
  - RedisGraph [20]
  - Neo4j [17]

2. Обзор

2.1. Используемые технологии

В качестве основной модели представления графов был выбран стандарт консорциума Всемирной Паутины RDF [3], ввиду широкой поддержки данного формата различными инструментами для работы с графами. Этот формат позволяет описывать отношения между ресурсами в виде "объект, предикат, субъект".

Вспомогательные инструменты написаны на языке python в связи с наличием поддерживаемых библиотек для работы с графами и грамматиками, таких как RDFLib [19], pyformlang [25].

Сайт проекта использует Github Pages и статический конструктор Jekyll, что позволяет не содержать отдельную документацию для сайта.

2.2. Существующие решения

Существует множество наборов данных для экспериментального анализа алгоритмов, реализующих регулярные запросы [18, 24, 8], которые представляют более узкий класс запросов, чем КС языки. Например, RBench [18] основан на RDF и использует генерацию бенчмарков, повторяющих структуру заданных в качестве эталона реальных данных. GSCALER [24] применяет похожий способ генерации для создания графовых БД.

Графы и грамматики, применяющиеся в исследованиях с использованием КС языков, представлены разрозненно, что создает сложность при поиске данных для экспериментального анализа нового алгоритма. Часто представлен набор данных только из одной практической области. Например, набор популярных онтологий, связанных с концепцией семантической паутины [21], можно найти в [4]. Исследования [13, 6] применяют для экспериментального анализа набор данных из репозитория [10], содержащий данные исключительно для статического анализа кода на Java.

2.3. Наработки исследовательской группы

Исследовательской группой в JetBrains Research был создан репозиторий CFPQ_Data, куда включены:

- Реальные данные, для которых используется same-generation query [7] — важный контекстно-свободный запрос, не являющийся регулярным.
  - Набор данных Geospecies [9], содержащий информацию о биологической иерархии и географии видов.
  - Набор RDF онтологий из [4].
  - Графы из открытой базы данных последовательностей белков Uniprot [22].
- Синтетические данные для особых случаев.
  - Предложенный в исследовании [12] крайний случай запроса для языка правильных скобочных последовательностей на графе с двумя циклами.
  - Разреженные графы для симуляции реальных разреженных данных.
  - Случай, когда результатом вычисления запроса на разреженном графе является полный граф.
  - Случайные безмасштабные сети, для генерации которых применяется модель Барабаши-Альберт [1].

3. Архитектура решения и описание реализации

Репозиторий CFPQ_Data содержит две основные директории: ./data с данными и ./tools с инструментами, упрощающими работу с набором данных. Данные собраны в группы, каждая из которых включает набор матриц и набор запросов, применяющихся с заданными графами.

3.1. Представление данных

Все графы, ранее представленные в простом txt формате, теперь используют XML сериализацию формата RDF, что решает проблему совместимости со сторонними инструментами для работы с данными.

Поскольку на данный момент не существует общепринятого формата представления КС запросов (например, язык Cypher, использующийся в Neo4j, поддерживает только регулярные запросы [5]), принято решение использовать следующее представление контекстно-свободных грамматик: строка 1 содержит нетерминалы грамматики, строка 2 — терминалы, а дальнейшие строки — продукции, в правых частях которых можно использовать регулярные выражения.

Такой формат позволяет, в сравнении с ранее использованным представлением в Нормальной Форме Хомского, сократить запись запроса и повысить его читаемость, например:

Listing 1: Грамматика в выбранном формате

```
S
a b
S -> (a S b)*
```

3.2. Документация

Реализован сайт репозитория на Github Pages. С помощью шаблона конструктора статических сайтов Jekyll добавлены метаданные набора

данных в формате JSON-LD, необходимые для его индексации в Dataset Search. Метаданные включают в себя ключевые слова для его поиска, информацию о его создателях, лицензии.

В README и на сайте проекта описан формат представленных данных, добавлены описание всех графов и грамматик и ссылки на использующие их исследования. Для проверки корректности алгоритмов включен CSV файл с информацией о количестве путей, порождаемых нетерминалами грамматик на представленных данных.

3.3. Загрузка данных в графовые БД

В директорию ./tools репозитория добавлен инструмент для загрузки данных в RedisGraph с помощью библиотек Redis и RedisGraph на python. При необходимости загрузки набора графов в Neo4j предлагается использование уже существующего плагина NSMNTX [16], позволяющего импортировать RDF графы в БД.

3.4. Преобразование представленных запросов в НФХ

Создан инструмент, позволяющий преобразовывать запросы в выбранном формате к Нормальной Форме Хомского следующим образом:

1. В первую очередь обрабатываются регулярные выражений. Рассмотрим один шаг алгоритма их обработки на продукции в виде $S \to regex$. Регулярное выражение в правой части преобразуется в ДКА, из переходов которого происходит построение новых продукций грамматики модификацией алгоритма, описанного Дж. Хопкрофтом [14]:

    - Для каждого перехода ДКА из состояния A в состояние B по символу c добавляются продукции в виде $A \to cB$, где A и B это нетерминалы, соответствующие состояниям ДКА. Для нетерминалов, соответствующих конечным состояниям ДКА, добавляются продукции $A \to \epsilon$.

- Отличие от стандартного алгоритма заключается в том, что нетерминал Q, соответствующий начальному состоянию ДКА, не становится стартовым нетерминалом. Вместо этого в грамматику добавляется продукция $S \to Q$, соединяющяя полученные продукции с грамматикой в построении.

2. Полученная в результате предыдущих шагов КС грамматика преобразуется в Нормальную Форму Хомского при помощи библиотеки pyformlang [25].

Заключение

В рамках работы над проектом были выполнены следующие задачи:

- Унифицировано представление данных.
  - Все графы представлены в XML сериализации формата RDF.
  - Разработан человекочитаемый формат запросов.
- Создан инструмент, позволяющий получать из запросов в выбранном формате грамматики в Нормальной Форме Хомского [2].
- Создан сайт проекта, индексируемый в Dataset Search [11], в результате чего проект уже появляется в поисковой выдаче.
- Добавлена документация с описанием представленных графов и грамматик, контрольными значениями вычисления запросов и ссылками на исследования, использующие эти данные для экспериментального анализа.
- Предоставлена возможность загрузки данных в следующие графовые БД:
  - RedisGraph [20]
  - Neo4j [17]

Дальнейшие планы по работе над набором данных включают в себя:

- Добавление в CFPQ_Data графов и КС запросов, использующихся для анализа происхождения данных [15].
- Поиск других областей и задач, из которых можно собрать новые данные.

Список литературы

[1] Albert Réka, lászló Barabási Albert. Statistical mechanics of complex networks // Rev. Mod. Phys. — P. 2002.

[2] Chomsky Noam. On certain formal properties of grammars // Information and Control. — 1959. — Vol. 2, no. 2. — P. 137 – 167. — URL: http://www.sciencedirect.com/science/article/pii/S0019995859903626.

[3] Consortium World Wide Web. RDF Primer. — 2004. — URL: https://www.w3.org/TR/rdf-primer/ (online; accessed: 12-12-2019).

[4] Context-free path queries on RDF graphs / X. Zhang, Z. Feng, X. Wang et al. // International Semantic Web Conference / Springer. — 2016. — P. 632–648.

[5] Cypher: An Evolving Query Language for Property Graphs / Nadime Francis, Andrés Taylor, Alastair Green et al. — 2018. — 05. — P. 1433–1445.

[6] Dietrich Jens, Hollingum Nicholas, Scholz Bernhard. Giga-scale Exhaustive Points-to Analysis for Java in Under a Minute // Proceedings of the 2015 ACM SIGPLAN International Conference on Object-Oriented Programming, Systems, Languages, and Applications. — OOPSLA 2015. — New York, NY, USA : ACM, 2015. — P. 535–551. — URL: http://doi.acm.org/10.1145/2814270.2814307.

[7] Foundations of Databases: The Logical Level / Ed. by Serge Abiteboul, Richard Hull, Victor Vianu. — 1st edition. — Boston, MA, USA : Addison-Wesley Longman Publishing Co., Inc., 1995. — ISBN: 0201537710.

[8] GMark: Schema-Driven Generation of Graphs and Queries / Guillaume Bagan, Angela Bonifati, Radu Ciucanu et al. // IEEE

Transactions on Knowledge and Data Engineering. — 2016. — 11. — Vol. PP.

[9] Geospecies. — URL: https://old.datahub.io/dataset/geospecies/ (online; accessed: 15-12-2019).

[10] Gigascale. — URL: https://bitbucket.org/jensdietrich/gigascale-pointsto-oopsla2015/ (online; accessed: 15-12-2019).

[11] Google Dataset Search. — URL: https://datasetsearch.research.google.com/ (online; accessed: 14-05-2020).

[12] Hellings Jelle. Querying for Paths in Graphs using Context-Free Path Queries. — 2015. — 1502.02242.

[13] Hollingum Nicholas, Scholz Bernhard. Cauliflower: a Solver Generator for Context-Free Language Reachability // LPAR-21. 21st International Conference on Logic for Programming, Artificial Intelligence and Reasoning / Ed. by Thomas Eiter, David Sands. — Vol. 46 of EPiC Series in Computing. — EasyChair, 2017. — P. 171–180. — URL: https://easychair.org/publications/paper/bnVq.

[14] Hopcroft John E., Motwani Rajeev, Ullman Jeffrey D. Introduction to automata theory, languages, and computation. — Pearson®, 2014.

[15] Miao Hui, Deshpande Amol. Understanding Data Science Lifecycle Provenance via Graph Segmentation and Summarization // 2019 IEEE 35th International Conference on Data Engineering (ICDE). — 2019. — P. 1710–1713.

[16] Neo4j RDF & Semantics toolkit. — URL: https://neo4j.com/labs/neosemantics-rdf/ (online; accessed: 14-05-2020).

[17] Neo4j graph database. — URL: https://neo4j.com/neo4j-graph-database/ (online; accessed: 13-12-2019).

[18] Qiao Shi, Özsoyoglu Z. Meral. RBench: Application-Specific RDF Benchmarking // SIGMOD '15. — 2015.

[19] RDFLib. — URL: https://github.com/RDFLib/rdflib (online; accessed: 14-05-2020).

[20] RedisGraph graph database. — URL: https://oss.redislabs.com/redisgraph/ (online; accessed: 13-12-2019).

[21] Semantic Web. — URL: https://www.w3.org/standards/semanticweb/ (online; accessed: 17-12-2019).

[22] Uniprot. — URL: https://www.uniprot.org/ (online; accessed: 17-12-2019).

[23] Yannakakis Mihalis. Graph-Theoretic Methods in Database Theory // Proceedings of the Ninth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems. — PODS '90. — New York, NY, USA : Association for Computing Machinery, 1990. — P. 230–242. — URL: https://doi.org/10.1145/298514.298576.

[24] Zhang J. W., Tay Y. C. GSCALER: Synthetically Scaling A Given Graph // EDBT. — 2016.

[25] pyformlang. — URL: https://pypi.org/project/pyformlang/ (online; accessed: 14-05-2020).