LOD Surfer API: クラス間関係を用いたLODからの情報抽出 Web API

LOD Surfer API: A Web API for LOD Surfing Using Class-Class Relationships

山口敦子[1]　小林紀郎[2]　臼田大輝[3]　桝屋啓志[3]　山本泰智[1]　古崎晃司[4]

Atsuko Yamaguchi[1], Norio Kobayashi[2], Taiki Usuda[3], Hiroshi Masuya[3], Yasunori Yamamoto[1] and Kouji Kozaki[4]

[1]情報・システム研究機構 ライフサイエンス統合データベースセンター
[1]Database Center for Life Science, ROIS
[2]理化学研究所 情報基盤センター
[2] Advanced Center for Computing and Communication, RIKEN
[3]理化学研究所 バイオリソースセンター
[3] BioResource Center, RIKEN
[4]大阪大学 産業科学研究所
[4] The Institute of Scientific and Industrial Research, Osaka University

概要: 生命科学分野ではLinked Open Data (LOD)が急速に普及しつつあり，これまで数多くのデータベースがLOD上で公開されている．これらのデータベースを有効に活用するためには，公開されているデータベースの物理的位置に関わらず,複数のデータベースから自在にデータを切り取り，組合わせて利用できることが望ましい．LOD上で公開されるデータベースはSPARQLエンドポイントと呼ばれるクエリ記述言語による検索が可能なウェブ API と共に提供されることが多い．SPARQLクエリ記述言語には，SERVICE句を用いた複数のエンドポイント検索の仕組みが用意されているが，この機能を利用するためには，どのエンドポイントにユーザが欲しいデータがどのような形式で格納されているかを予め知っておく必要がある．一方，著者らはこれまでSPARQLクエリの記述支援のため，SPARQL Builder Metadataとよぶ各エンドポイントのデータスキーマに関するメタデータを収集してきた．そこで，著者らは，これらのメタデータからクラス間関係をベースとした一つのグラフを構築することで，LOD 上のデータを柔軟に切り取る手法を提案する．そして，そのプロトタイプとして実装したLOD Surfer APIを紹介し，その利用について議論する．

1. はじめに

1.1 背景

日々生産される生命科学のデータは巨大かつ多岐にわたる．実験によって得たデータを解釈し，生物学的に新たな知識を得るためには，これらのデータ全体を統合し，自在に組合わせることが必須となってきている．このような状況のもと，数多くの研究施設が生産した生命科学データのRDF化をすすめ，Linked Open Data (LOD)として公開してきている．LOD上でデータベースを公開する際には,RDFの標準的な検索言語であるSPARQLを用いた検索が可能なウェブAPIである，SPARQLエンドポイントが提供されることが多い．その結果,LOD上のSPARQLエンドポイントは次々と新たに提供されており，ユーザがその全体像を理解するのは困難である．

LOD 上のデータを組み合わせて検索するための既存の仕組みとして,SPARQL言語のSERVICE句を用いたフェデレート検索が挙げられる．フェデレート検索では,SERVICE句によって部分クエリを投げるSPARQLエンドポイントを指定することができ，その結果，複数のSPARQLエンドポイントを一つのクエリで検索することができる．しかしながら，

SERVICE 句を用いたクエリを書くためには，元のクエリにおいてどの部分クエリがどの SPARQL エンドポイントに含まれている可能性があるか熟知している必要がある．また，SERVICE 句を用いなくとも，部分クエリを投げる SPARQL エンドポイントを ASK クエリなどで判定し，フェデレート検索を行うシステムもこれまで数多く開発されてきているが[1,2]，生命科学のような巨大なデータに用いるにはスケーラビリティの面で問題がある．

一方，著者らはセマンティックウェブ技術にあまり詳しくないユーザを対象に，SPARQL クエリ生成支援システム SPARQL Builder を開発し，2013 年 10 月からサービス運用を行ってきた[3,4]．その際，高速にクラス間関係提示を行うため，予め LOD 上の SPARQL エンドポイントをクロールし，SPARQL エンドポイントで提供されるデータのデータスキーマを記述するメタデータを取得し蓄積してきた．

そこで，これらのメタデータを利用し，LOD 全体の構造，および，どのエンドポイントから必要な情報が得られるかをユーザが把握する必要がなく，柔軟に LOD からデータを切り出すことを可能とする情報基盤 LOD Surfer を提案する．本発表では特に，LOD Surfer を実現するにあたり，必要な機能と課題を検討するために，プロトタイプ実装を行ったウェブ API，LOD Surfer API について紹介する．

1.2 SPARQL Builder Metadata

SPARQL Builder (図 1)とは，SPARQL 言語の知識がなくとも，また対象データセットの構造を知らなくても，対話的な GUI を介して SPARQL クエリを生成することができることを目指して開発されたウェブ上のサービスである[4].

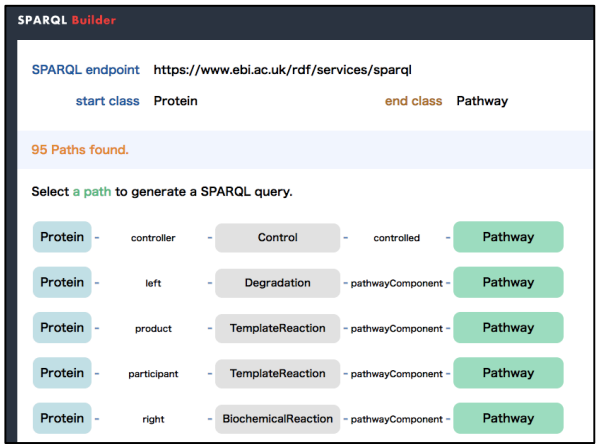


図 1: SPARQL Builder

SPARQL Builder はインスタンス間がどのようなプロパティでつながっているかというクラス間の関係をベースとして SPARQL クエリの記述補助を行う．ユーザはまず，入力クラスと出力クラスをそれぞれクラスのリストから選ぶ．入出力の二つのクラスが確定すると，それらのクラス間の関係でデータ内に含まれるものがユーザに提示される．ここで，ユーザが望むクラス間の関係は，データ内ではどのプロパティを用いて記述されているか分からないことに加え，複数のトリプルを用いた多段の関係により記述されている可能性もある．そこで，ユーザの欲しい関係を提示できるよう，入出力クラス間の多段の関係，つまりクラス間関係のパスを提示する．現在の SPARQL Builder では，最大 4 段のクラス間関係パスをすべて提示する．ユーザがパスを一つ選ぶと，そのパスのクラス間関係に対応する SPARQL クエリが自動生成される．生成された SPARQL クエリはそのまま検索へ利用することができ，また，生成された SPARQL クエリを編集して利用することも可能なシステムとなっている．

ユーザにデータセットやクラスの情報を提示するデータ，および，パスの計算を高速に行うために必要なデータは，事前に SPARQL エンドポイントから取得し蓄積しておくことが望ましい．この目的で設計されたメタデータが SPARQL Builder Metadata (SBM)である[5].

SBM のスキーマ仕様は，VoID [6]および SPARQL 1.1 Service Description [7]の拡張として定義されており，大まかには，SPARQL エンドポイント→エンドポイントに含まれるデータセット→データセットのメタデータの階層構造になっており，各メタデータ部分にはクラスリスト，プロパティリスト，クラス-プロパティ-クラス関係，さらにそれらに関連するトリプル数等の統計情報が含まれる．SBM について詳しくは，[5]を参照されたい．

SPARQL Builder を運用するにあたり，生命科学分野の 43 の SPARQL エンドポイントから 76 のデータセットのメタデータを取得し蓄積している．メタデータ取得は主に，SPARQL エンドポイントに対し，一連の SPARQL クエリを投げ，その結果から SBM を構成する．SBM 取得プログラムは公開されており，https://github.com/sparqlbuilder/metadata から入手可能である．

2. LOD Surfer

2.1 LOD Surfer 開発概要

LOD Surfer は，LOD から自分のデータに紐付いた

データの中から，欲しいデータがどの SPARQL エンドポイントに含まれるか気にすることなく，興味がある部分を柔軟に切り出すことを可能とすることを目指して設計されたシステム基盤である．対象データはウェブ上に分散しているため，LOD Surfer を実現するためには，データの所在を自動的に見つけ，さらにそのデータを適切かつ現実的な時間で取得する必要がある．まず，データが所在する SPARQL エンドポイントを見つけるためには，先述の SBM が利用可能である．SPARQL エンドポイントから必要なデータを適切かつ現実的な時間で取得することを担保するためには，Umaka-Yummy Data[8]を利用する．Umaka-Yummy Data は，生命科学データを中心とした SPARQL エンドポイントのリストに対し，毎日アクセスを行い，サービスの死活，更新頻度，VoID 等のメタデータ提供状況，クエリ返答速度などを監視し，その結果をスコア付けするシステムである．このシステムのウェブ API を利用し，稼働率が低い SPARQL エンドポイントやクエリ返答速度が遅い SPARQL エンドポイントをフィルタすることで，検索先のサービスが返答せず，データが取得できないことを回避する．

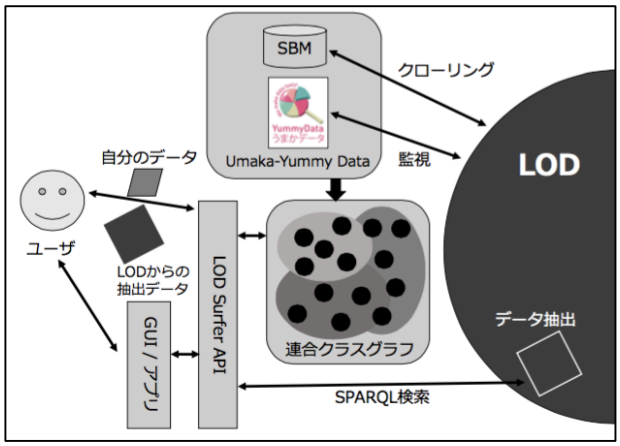


図 2: LOD Surfer システム構成設計

図 2 は LOD Surfer のシステム構成設計の概要である．Umaka-Yummy Data によりフィルタされクエリ応答が担保された SPARQL エンドポイントのリストに対し，対応する SBM 形式のメタデータを利用し，LOD 全体のクラス間関係を示すグラフ，連合クラスグラフを構築する．ユーザが LOD Surfer API を通じて自分のデータを入力すると，そのデータのクラスを始点としたクラス間関係パスを，連合クラスグラフを用いて計算し，到達可能なクラスのリストを示す．それらのクラスのリストから，ユーザが興味をもったクラスを指定すると，クラス間パスからフェデレート検索クエリを生成し，クエリを LOD 上で実行し，その結果をユーザへ返すことができる．

計算機に詳しくなく，LOD Surfer API を直接利用することが難しいユーザに対しては GUI やアプリケーションソフトウェアを通じて API を利用するように開発を進める予定である．

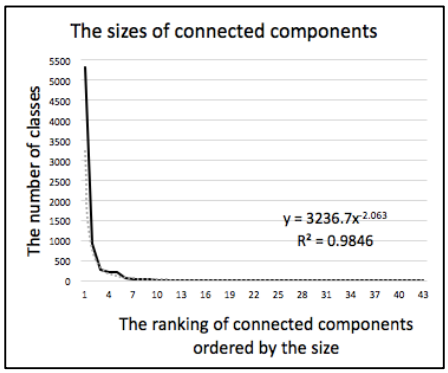
2.2 連合クラスグラフと連結成分計算

先述の SPARQL Builder においては，クラス間パスを高速に計算するために，SPARQL エンドポイントごとに，頂点をクラス，辺をクラス間の関係としたグラフ，クラスグラフを構築し利用した[4]．これらのクラスグラフに対し，同じクラスに対応する頂点を重ねることで，一つの大きなクラスグラフにすることができる．これを連合クラスグラフと呼ぶ．LOD Surfer におけるフェデレート検索クエリ構築もクラス間関係をベースとするため，効果的なフェデレート検索の設計のためには連合クラスグラフの構造の解析が必要である．そこで，現在 SPARQL Builder で利用中の 43 の SBM を利用し，連合クラスグラフを構築し，その解析を試みた．特に，クラス間の到達可能を考慮するために連結成分を中心に解析を行った．ちなみに，グラフの連結成分とは，グラフを互いにパスを持つ極大頂点集合に分割したものである．言い換えるならば，同じ連結成分内のクラス間には何らかのパスがあるといえる．

SBM から構築した連合グラフは，頂点数 14,147，辺 18,738 のグラフとなった．このグラフの連結成分を計算し，その頂点数(=クラス数)の分布を調べた．連結成分の数は 6,857 個であり，そのうち，6,814 個が孤立点であった．一方，最大の連結成分は 5,327 個のクラスを含み，孤立しない 7,333 個のクラスのかなりの部分を最大の連結成分が占めることがわかる．

図 3 は，孤立点でない連結成分について，連結成分の頂点数をサイズが大きいものから順に示したものである．上下二つのグラフは Y 軸に関し，上は線形軸，下は対数軸となっている．分布は冪乗則に従うように見え，事実，最小二乗法でフィッティングすると $y=3236.7x^{-2.063}$ に対し，その決定係数 R^2 は 0.9846 と十分 1 に近い値を取る．

このことから，(1)連合クラスグラフは連結成分を予め計算し，連結成分ごとに別に管理することで，探索空間を大幅に減らすことができる，(2)孤立点となるクラスが多いので，上位クラスでまとめる工夫が必要である，ということができる．



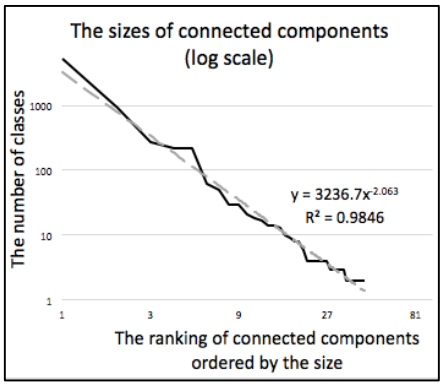


図 3: 連合クラスグラフの連結成分サイズの分布

3. LOD Surfer API

3.1 API 設計と実装

LOD Surfer API はユーザがクラス間関係を利用して LOD から効率良くかつ柔軟にデータを切り出すためのウェブ API である．この API を利用して，対象となる SPARQL エンドポイント一覧，クラス一覧，あるクラスから到達可能なクラスの一覧，二つのクラスに対し，連合クラスグラフ上のクラス間パスのリスト，クラス間パスに対し，SERVICE 句付きのフェデレート検索クエリを出力することができる．

これらの機能を実現するウェブ API を下記のように設計した．

- /eplist?class1={class1 URI}
  class1 をデータ内に含む SPARQL エンドポイントのリストが JSON 形式で出力される．class1 が指定されない場合，全ての対象 SPARQL エンドポイントのリストが出力される．
- /clist?class1={class1 URI}
  class1 から到達可能なクラスのリストが JSON 形式で出力される．class1 が指定されない場合，全てのクラスのリストが出力される．
- /path?class1={class1}&class2={class2}
  class1 から class2 へのパスのリストが JSON 形式で出力される．
- /sparql?path={path}
  path に対応する SERVICE 句付き SPARQL クエリがテキスト形式で出力される．

これらの機能は，連合クラスグラフおよびその連結成分を用いて高速に実行できる．例えば，到達可能性は連結成分を用いて瞬時に計算でき，パスも連結成分内のみに探索空間が限定されるため，現実的な時間で計算できる．

LOD Surfer API の高速性を確認するため，現在 SPARQL Builder で利用中の 43 の SBM に対し，プロトタイプを作成した．このウェブ API は直接 RDF 形式の SBM を利用するわけではなく，グラフアルゴリズムが適用しやすいように，一旦，連合クラスグラフを表す隣接リスト構造へ変換したものを利用した．さらに，効率性を上げるために連結成分を計算したうえで，成分ごとに頂点のハッシュ集合で保持した．これらの構造上で，上記の機能を持つウェブ API を作成した．

LOD Surfer API をプロトタイプ実装したものは http://lodsurfer.dbcls.jp/api/で試すことができる．例えば，全クラスリストは http://lodsurfer.dbcls.jp/api/clist で出力することができる．また，システムのコードは https://github.com/LODSurfer/lodsurfer-api から取得可能である．

3.2 LOD Surfer API 利用例

LOD Surfer API はクラス間関係をベースとした検索アプリケーションでの利用をターゲットにしている．そのため，少しでもプログラミングができれば，これを利用した LOD の検索アプリケーションは誰にでも簡単に作ることができる．

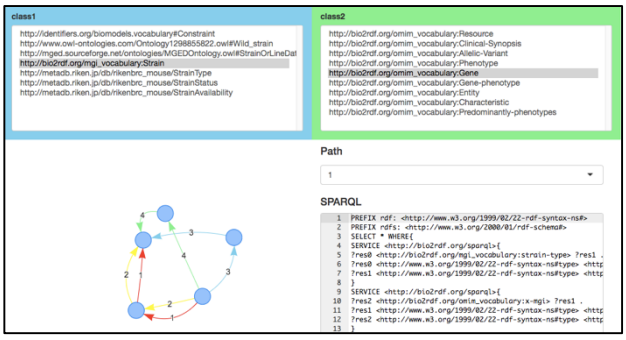


図 4: LOD Surfer API を利用した GUI 例

図 4 は LOD Surfer API を利用した，R 言語で作成

したアプリケーションの例である．クラス選択ペインが上部に，クラス関係表示ペインが左下に，パス選択と SERVICE 句付きの SPARQL クエリ表示ペインが右下に表示されている．この検索アプリケーションを用いて，クラス選択から SERVICE 句付きの SPARQL クエリ表示，さらには結果取得までスムーズに行えることを確認した．

4. まとめと考察

クラス間関係をベースとした一つのグラフを構築することで，LOD 上のデータを柔軟に切り取る手法を提案した．また，この手法のプロトタイプ実装として，SPARQL Builder 運用において収集した SBM を用い，連合クラスグラフを構築し，ウェブ API を開発した．

連合クラスグラフの解析結果より，インスタンスが少ないクラスや孤立したクラスは，より上位クラスで下位クラスをまとめたうえでクラス間関係を検索することが望ましい．しかしながら，現時点での SBM は rdfs:subClassOf への対応が十分ではなく，SBM の情報だけでは，下位クラスを適切に上位にまとめることが難しい．そのため，今後，SBM を LOD Surfer 向けに設計を直す必要があり，また，そのためのクローラシステムの開発も必須である．

また，今回の LOD Surfer API のプロトタイプ実装では，最終的にフェデレート検索クエリの出力で留まり，クエリの処理は他の SPARQL エンドポイントへ任せる設計になっている．今後，SBM 内の統計情報などを利用することにより，データの結合を効率的に行える順序を計算することができると思われる．その手法を使ったクラス間関係ベースのフェデレート検索システムの提案も考えたい．

謝辞

本研究は独立行政法人科学技術振興機構(JST)、バイオサイエンスデータベースセンター (NBDC) の助成，および科学研究費補助金基盤(C) 17K00434，17K00424，基盤(B) 17H01789 の助成による．

参考文献：

[1] Schwarte, A., Haase, P., Hose, K., Schenkel, R., Schmidt, M.: Fedx: A federation layer for distributed query processing on linked open data. In: 8th Extended Semantic Web Conference (ESWC 2011), LNCS 6644, 481–486 (2011)

[2] Wu, H., Yamaguchi, A., Kim, J. D.: Dynamic Join Order Optimization for SPARQL Endpoint Federation. In: 11th International Workshop on Scalable Semantic Web Knowledge Base Systems (SSWS 2015), CEUR Workshop Proceedings, vol. 1457, 48-62 (2015)

[3] Yamaguchi, A., Kozaki, K., Lenz, K., Wu, H., Kobayashi N.: An Intelligent SPARQL Query Builder for Exploration of Various Life-science Databases, The 3rd International Workshop on Intelligent Exploration of Semantic Data (IESD 2014), CEUR Workshop Proceedings 1279

[4] Yamaguchi, A., Kozaki, K., Lenz, K., Yamamoto, Y., Masuya, H., Kobayashi N.: Semantic Data Acquisition by Traversing Class-Class Relationships Over Linked Open Data, The 6th Joint International Semantic Technology Conference (JIST 2016), LNCS 10055, 136-151

[5] SPARQL Builder Metadata (Version Sep. 2015), http://www.sparqlbuilder.org/doc/sbm_2015sep/

[6] Describing Linked Datasets with the VoID Vocabulary, https://www.w3.org/TR/void/

[7] SPARQL 1.1 Service Description, https://www.w3.org/TR/sparql11-service-description/

[8] Yamamoto, Y., Yamaguchi, A., Splendiani, A.: YummyData: providing high-quality open life science data. Database, doi: 10.1093/database/bay022, 2018