D-018

クローラ型 Web 検索エンジンを利用した SPARQL Endpoint 発見手法
Discovering the SPARQL Endpoints by using Crawler-based Web Search Engine

瀬尾 崇一郎† 阪口 哲男‡
Soichiro SEO Tetsuo SAKAGUCHI

1. はじめに

近年，Web を通じたデータの公開・共有を目指すオープンデータの動きが盛んになってきており，Linked Open Data（以下，LOD）と呼ばれる公開方式が W3C によって推奨されている．この LOD においてはデータ公開に伴って SPARQL Endpoint と呼ばれる Web API が用意されることが多く，オープンデータの重要な利用手段となっている．

本論文では， Web 検索エンジンのクローラによって捕捉された既存の SPARQL Endpoint を，SPARQL Endpoint に特徴的な文字列による Web 検索によって取得することで，LOD の公開者および利用者を支援する手法を提案する．

1.1 RDF と LOD

RDF とは主語，述語，目的語から成るトリプルを用いて情報リソースを記述するためのデータモデルである．

この RDF によって構造化されたデータを Web 上でリンクして利用性を高めようとする実践的方法が Linked Data であり，この Linked Data に基づいたオープンデータが LOD と呼ばれる．

Linked Data においては URI によってリソースを一意に表現し，データ中には他の Linked Data 中の URI やリソースへのリンクを含むことで，外部データとの連携による利用性が高められている．

1.2 LOD 提供の方法である SPARQL Endpoint

SPARQL Endpoint とは，RDF データを格納するデータベース（以下，RDF ストア）が RDF データ用のクエリ記述言語である SPARQL によって記述した検索クエリを受け付けるための Web API である．

SPARQL クエリは SPARQL Endpoint の URL に基づいて HTTP リクエストによって送信され，指定した形式によって結果を取得できる．また，Web ブラウザから SPARQL Endpoint の URL にアクセスすると，SPARQL クエリの入力フォームを備えた Web ページ（以下，Web UI）が表示されることもある．

Virtuoso[1]などのオープンソースの RDF ストアや，AllegroGraph[2]などの商用の RDF ストアに RDF データを格納させると，Web UI を備えた SPARQL Endpoint として容易に運用できるようになる．このことから，多くの LOD がこれら RDF ストアを利用し SPARQL Endpoint を公開している．

一般に LOD の利用者は，RDF データをダウンロードして自らの RDF ストアに格納するか，公開されている SPARQL Endpoint に SPARQL クエリを送り結果を取得することで LOD を利用することになる．LOD 利用者から見た SPARQL Endpoint のメリットとして，利用する LOD のファイルをダウンロードすることなく SPARQL による検索が行え，LOD に追加や更新があった場合に手元の RDF ストアの同期を考えることなく利用できることがあげられる．

2. LOD と SPARQL Endpoint の発見とその課題

2.1 既存の LOD を発見する方法

既に公開されている LOD の存在を，その LOD の潜在的な利用者が把握することは，決して容易なことではない．

LOD 利用者が既存の LOD を発見する方法には，オープンデータ公開のためのポータルサイトを利用することがある．このポータルサイトとしては the Datahub[3]や，Data for Japan[4]などが該当し，オープンデータの内容に関する説明や，公開 URL などが集められ公開されている．これらは CKAN というオープンソースのポータルサイトソフトウェアを使用しており，登録されたデータに関するメタデータは CKAN API によって比較的容易に取得できる．ただし，LOD の公開者自身が情報を登録する必要があり，更に登録内容に変更があった場合には自ら更新を行う必要がある．

また，既存の RDF データを集積し，それらの内容に対する検索を可能にしている例に，Sindice[5]がある．Sindice はデータ公開者が予め登録している公開データを定期的に巡回することで収集と更新を自動的に行い，それらに全文検索的な解析を行うことで，データ利用者が検索できるようにしている．ただしデータ公開者が Sindice を利用するためには，Sindice が規定するデータの公開と更新に関する条件を満たした上で，データ公開者が Sindice に対し登録を依頼する必要がある．

2.2 既存の SPARQL Endpoint を発見する方法

ある LOD の SPARQL Endpoint を利用するには，SPARQL Endpoint が設置されている URL を把握する必要がある．

the Datahub ではオープンデータの登録内容として SPARQL Endpoint の URL を登録でき，提供フォーマットの種類として明記することができる．これを条件として検索することによりオープンデータの利用者は SPARQL Endpoint を持つ LOD を発見できる．しかし登録情報の未更新やサービス終了等の理由から，もはや有効ではなくなってしまった SPARQL Endpoint の URL が the Datahub には多く登録されている．

† 筑波大学大学院図書館情報メディア研究科 Graduate School of Library Information and Media Studies, University of Tsukuba
‡ 筑波大学 図書館情報メディア系/知的コミュニティ基盤研究センター Research Center for Knowledge Communities, Faculty of Library, Information and Media Science, University of Tsukuba

Copyright © 2014 by
The Institute of Electronics, Information and Communication Engineers and
Information Processing Society of Japan All rights reserved.

また，LOD を公開しているサイト上において，該当する LOD に対するドキュメントを人が読み取ることで SPARQL Endpoint の URL を取得する方法も考えられる．この場合は最新の情報を取得できる可能性が高いが，まず LOD が公開されているサイトを発見し，サイト内から LOD に関するドキュメントを探索した後，更にドキュメントから SPARQL Endpoint の URL を発見するという一連の作業が必要であり，利用者に負担となってしまう．

Paulheim らは，LOD の RDF データ中に存在する URI から，その URI に責任を持って公開している LOD の SPARQL Endpoint を発見するシステムを研究している[6]．これによって，RDF データ中に既知でない URI を使用したリソースが出現したとしても，その URI に関する有効な問い合わせが可能な SPARQL Endpoint を発見できる．Paulheim らは 2 種類のアプローチを取っており，1 つは URI の文字列から LOD のメタデータである VoID データ[7]が保管されている URL を推定し，これによって取得した VoID データに記述されている SPARQL Endpoint の URL を取得するアプローチである．しかしこれには該当 SPARQL Endpoint に VoID データが用意されていることが前提条件となる他，Paulheim らが想定するようなルールに沿った URL に公開者が VoID データを設置するとは限らないという問題があるため，確実性に欠ける．もう 1 つのアプローチは，その URI を扱っている SPARQL Endpoint をカタログデータの中から探し出すものであるが，このカタログには the Datahub のデータを使用しているため，先述したデータ公開者の情報登録と更新の必要性や，有効ではなくなった登録情報などといった問題が付きまとう．

以上のことから，公開者の自発的な登録に依存せず，利用者にも大きな負担をかけない，より自動化された SPARQL Endpoint 発見手法が必要であると考えられる．

3. クローラ型 Web 検索エンジンを利用した SPARQL Endpoint 発見手法

本研究では，Web 上に存在する SPARQL Endpoint を自動的に発見，収集することにより LOD の公開者並びに利用者を支援することを目的として，クローラ型 Web 検索エンジンを用いた SPARQL Endpoint の発見手法を提案する．

クローラ型 Web 検索エンジンとは，Web ページのリンク関係を辿ることで Web ページを自動的に収集する Web 検索エンジンのことであり，Google や Bing などがこれに該当する．一般的に，LOD 公開者によって Web 上のある URL に設置された SPARQL Endpoint は，LOD 利用者への告知のために何らかの形で他の Web ページから参照される．特に Web UI を持つタイプの SPARQL Endpoint においては，Web ブラウザからのアクセスおよび SPARQL クエリの構築・送信と結果の取得が可能であるため，告知用の Web ページから直接リンクを張られることが多い．従って，Web 検索エンジンのクローラが SPARQL Endpoint の Web ページを収集していることは，十分に考えられる．

また，Virtuoso 等の RDF ストアを使用して構築した SPARQL Endpoint の場合，同じ RDF ストアを使用した SPARQL Endpoint の間で Web UI の文書構造が類似するため，それらに特徴的な文字列が含まれていると考えられる．この特徴的な文字列を調べ，検索クエリとして Web 検索エンジンで検索することで，公開者の自発的な登録に依存せずに SPARQL Endpoint を発見できると考えられる．

クローラ型 Web 検索エンジンを利用することには，次のようなメリットがある．第一に，データ公開者が自らポータルサイトに登録をせずとも，公開者の管理するサイトに SPARQL Endpoint を設置しリンクを張るだけで発見が可能になる．第二に，SPARQL Endpoint の URL 等に変更があった場合，データ公開者が登録内容の変更を行わずとも利用者が変更後の SPARQL Endpoint を捉えられる．第三に，公開方法に依存しない機械的な発見手法を用いることによって，LOD を公開するサイトや登録するポータルサイトの知名度などによる発見性の差が小さくなり，データ公開者が公開方法の選択をより自由に行うことができる．

4. 調査と実験

4.1 Web UI を持つ SPARQL Endpoint の調査

検索エンジンのクローラが SPARQL endpoint を取得するためには Web UI が SPARQL Endpoint に用意されている必要があり，SPARQL クエリを受け付ける URL のみが用意されている SPARQL Endpoint に対してはクローラがキャッシュを残さないため，3 節で述べた SPARQL Endpoint 発見手法は適用できない．

そこで，まず Web UI を持たない SPARQL Endpoint がどの程度の割合で存在しているかについて調査を行った．調査は the Datahub から取得した 465 件の SPARQL Endpoint の URL を対象とし，それら URL に Web ブラウザからアクセスしたときに，Web UI が表示される SPARQL Endpoint，Web UI は表示されないが SPARQL クエリは受け付ける SPARQL Endpoint，無効となった SPARQL Endpoint の URL が，それぞれ何件あるかを調べた．結果を表 1 にまとめる．

表 1　2014 年 2 月 13 日に Datahub に登録されていた SPARQL Endpoint の Web UI に関する内訳

Web UI あり	Web UI なし	無効な URL	合計
221	45	199	465

（2014 年 2 月 20 日調査）

the Datahub に登録されている SPARQL Endpoint の中で，現在有効な Endpoint は 80%以上が同じ URL で Web UI を表示させることができる．従って，Web UI を手掛かりにした発見手法により，全てではないが多くの SPARQL Endpoint を発見できる可能性があると言える．

4.2 Google 検索を利用した SPARQL Endpoint の発見実験

クローラ型 Web 検索エンジンを利用した SPARQL Endpoint 発見手法の妥当性を確かめるため，代表的なクローラ型検索エンジンである Google 検索を利用することで Web UI および SPARQL Endpoint を発見できるかを実験によって検証した．

発見対象とする Web UI としては，RDF ストアである Virtuoso，AllegroGraph，そして SPARQL Endpoint 構築ツールである SPARQLer[8]によって生成された Web UI を設定す

Copyright © 2014 by
The Institute of Electronics, Information and Communication Engineers and
Information Processing Society of Japan All rights reserved.

る．これら 3 種のツールによって生成された Web UI に対する Google のキャッシュを調べた結果，表 2 で示すような文字列がそれぞれ特徴的であると判断し，これを検索クエリとして使用するものとする．

表 2　Web UI と対応する検索クエリ

ツール	特徴的な文字列
Virtuoso	“Virtuoso SPARQL Query Editor”
AllegroGraph	“AllegroGraph Webview is”
SPARQLer	“SPARQLer – General purpose processor”

これら文字列をダブルクォーテーションで囲むことで Google 検索におけるフレーズ検索のクエリとする．また「filter=0」のパラメータ設定を行うことで，類似する検索結果の除外機能を使用せずに検索を行う．以上の条件による検索結果から，3 種の検索クエリそれぞれの結果の上位 40 件，合計 120 件の検索結果を取得し，SPARQL Endpoint である Web ページが何件含まれていたかを調べる．Web ページが SPARQL Endpoint であるかどうかの判別は，簡単な SPARQL クエリを送信することによって行う．

表 3　実験で取得した Web ページにおける SPARQL Endpoint の内訳

SPARQL Endpoint と判別されたページ数	SPARQL Endpoint と判別されなかったページ数	合計
47	73	120

表 3 の通り，3 種の Web UI を対象にした検索によって，47 件の SPARQL Endpoint を取得することができている．

また，the Datahub に登録されている SPARQL Endpoint は，登録されていない SPARQL Endpoint よりも検索エンジンのクローラに捕捉されやすいと考えられる．そこで，本研究の手法によって the Datahub に登録のある SPARQL Endpoint だけでなく，登録のない SPARQL Endpoint についても発見できているかを調べた．表 3 において SPARQL Endpoint と判別された Web ページ 47 件それぞれの URL について，the Datahub における登録の有無を確認した結果が，表 4 である．

表 4　取得した SPARQL Endpoint における the Datahub 登録数の内訳

the Datahub に登録されていた SPARQL Endpoint 数	登録の無かった SPARQL Endpoint 数	合計
23	24	47

表 4 の通り，Datahub に登録のない 24 件の SPARQL Endpoint が，今回の検索実験によって取得できている．このことから，本研究の手法は確かに the Datahub で登録のある SPARQL Endpoint と，the Datahub に未登録である SPARQL Endpoint の両方を発見することができると言える．

4.3 関連研究

本研究ではクローラ型検索エンジンが取得したキャッシュを利用して SPARQL Endpoint の Web UI を取得しているが，関連研究としてクローラが収集した Web ページについて WebAPI であるか否かを判別する Steinmetz らの研究がある[9]．この研究では，取得した Web ページ内の出現語を利用し SVM によって分類する自動分類アプローチと，Web ページの URL やタイトル，本文などといったそれぞれの箇所における WebAPI に特有の特徴語を見つけ出そうとする頻出語アプローチの 2 つが試みられているが，論文中ではどちらも未だ試験段階の未完成な手法であるとされており，これらについて検出率などといった具体的な評価の結果は記されていない．

本研究は SPARQL Endpoint を対象とし，URL にアクセスすると表示される Web UI を判別に利用することで，SPARQL Endpoint の検出を実現させている．また，本研究は既存のクローラの機能に基づき発見を行う点が，新型クローラの開発を想定している Steinmetz らの研究とは異なっている．

5. 今後の課題

本論文では，SPARQL Endpoint の URL と Web UI の URL が一致していることが，取得できる SPARQL Endpoint の条件となっている．しかし SPARQL Endpoint には，4.1 で述べたように Web UI が存在しない場合や，Web UI は用意されているが SPARQL Endpoint の URL とは一致しない場合がある．本研究の手法では Web UI を持たない SPARQL Endpoint については取得することができず，また SPARQL Endpoint と URL が一致しない Web UI が検索エンジンによって取得された場合にも，対応する SPARQL Endpoint の URL を取得することは未だできていない．これらの問題への対応が，今後の課題である．

謝辞

本研究を進めるにあたって貴重なご意見を頂きました，筑波大学図書館情報メディア系の杉本重雄教授，森嶋厚行教授，永森光晴講師に，深く感謝致します．

参考文献

[1] Openlink Viertuoso Universal Server. http://virtuoso.openlinksw.com/
[2] AllegroGraph RDFStore Web 3.0’s Database. http://franz.com/agraph/allegrograph/
[3] Welcome – the Datahub. http://datahub.io/
[4] Data for Japan. http://dataforjapan.org/
[5] Sindice – the semantic web index. http://sindice.com
[6] Heiko Paulheim, Sven Hertling, “Discoverability of SPARQL Endpoints in Linked Open Data”, CEUR Workshop Proceedings, Vol.1035 (Proceedings of the ISWC 2013 Posters & Demonstrations Track), 2013, pp.245-248
[7] Describing Linked Datasets with the VoID Vocabulary, http://www.w3.org/TR/void/
[8] SPARQLer – An RDF Query Server. http://sparql.org/
[9] Nathalie Steinmetz, Holger Lausen, Manuel Brunner, “Web Service Search on Large Scale”, Service-Oriented Computing, Lecture Notes in Computer Science, Vol.5900, 2009, pp.437-444.

Copyright © 2014 by
The Institute of Electronics, Information and Communication Engineers and
Information Processing Society of Japan All rights reserved.