Web ページとしての類似性を利用した Linked Data リポジトリの自動収集手法

筑波大学
図書館情報メディア研究科

2015 年 1 月

瀬尾　崇一郎

1. はじめに

近年，Web を通じたオープンデータの動きが盛んになってきている．このオープンデータについて，Linked Data と呼ばれる実践的方法によって行う Linked Open Data と呼ばれる公開方式が W3C によって推奨されており，様々な機関や団体が自らの Linked Data を提供するための Linked Data リポジトリを公開している．利用者が Linked Data リポジトリを発見するための方法として現在主として使われているのはデータカタログサイトと呼ばれるポータルサイトである．これにはリポジトリ情報の登録や更新を人手の作業に頼っているという問題点があり，リポジトリ数が世界中で増加し続ければ対応できなくなってしまうことが予想される．そこで本研究では，Web 上で公開されている Linked Data リポジトリを自動的に収集する手法を提案する．

本論文においては，まず第 2 章で Linked Data リポジトリにおける背景と現状の問題について述べる．第 3 章では Web ページとしての類似性を利用した Linked Data リポジトリの自動収集というアイデアの概要を説明し，第 4 章ではそれを実現するための手法についてより詳細に述べる．第 5 章では実際の Linked Data リポジトリと検索エンジンを使用して，提案する Linked Data リポジトリ収集手法の収集能力を評価し，考察を行う．

2. 背景

2.1 RDF と Linked Open Data

RDF（Resource Description Framework） [1]とはウェブ上のリソースに関する情報を論理的に表現するデータモデルであり，それを記述するための言語体系である [2]．RDF では主に主語，述語，目的語から成るトリプルを用いて，記述対象を表現する．図 1 は，ある URI によって示される概念が“瀬尾　崇一郎”というラベルを持っていることを，URI を主語，ラベル関係を表現する語彙である rdfs:label を述語，“瀬尾　崇一郎”という文字列を目的語とするトリプルによって表現した，簡単な RDF データの例である

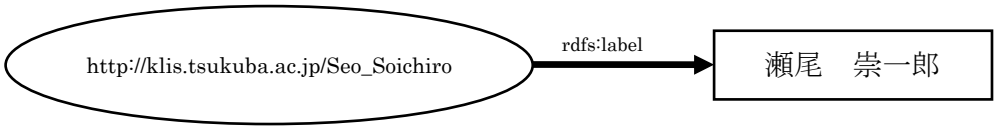

図 1　RDF データの例

この RDF によって構造化されたデータ同士を Web 上でリンクして，相互に利用性を高めようとする実践的方法が Linked Data である．Linked Data では他所の RDF で定義された概念を主に目的語として参照することで，参照先の RDF データと結合させて利用することを可能にしている．この Linked Data は政府や自治体で自らの収集したデータを公開するオープンデータの活動において頻繁に利用されており，そのようなオープンデータは特に Linked Open Data（以下，LOD）と呼ばれている．

2.2 Linked Data リポジトリと SPARQL Endpoint

一般的に Linked Data は広く一般の利用者に利用されることを目的とし，Web を通じて世界中に公開されるものである．ここで公開者が Linked Data を利用者に提供するために用意する Linked Data 公開用の Web サイトを，本研究では Linked Data リポジトリと呼ぶ. Linked Data リポジトリが Linked Data を提供する方法には, Linked Data である RDF データを, xml 形式や n-triple 形式で表現したデータファイルとしてユーザがダウンロード可能な状態にして置くほか，自らが扱う Linked Data を RDF データ用のデータベース（以下，RDF ストア）に格納した上で Web API を用意し，ユーザのクエリを直接受け付け，該当する Linked Data を提供する方法がある．このような Web API としては RDF データ用のクエリ記述言語 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 が存在しており，Linked Data を利用する際にはこれを用いることが主流となっている．このような Web API サービスは，SPARQL Endpoint と呼ばれている．

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 [3]などのオープンソースの RDF ストアや, AllegroGraph [4]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 Linked Data リポジトリがこれら RDF ストアを利用した SPARQL Endpoint を公開している．

一般に Linked Data 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, Linked Data リポジトリで公開されている SPARQL Endpoint に SPARQL クエリを送り結果を取得することで Linked Data を利用することになる．利用者から見た SPARQL Endpoint のメリットとしては，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．

2.3 データカタログサイトとその問題

ある Linked Data リポジトリの SPARQL Endpoint を利用するには，当然ながらその Linked Data リポジトリの存在を知った上で，その Linked Data リポジトリに用意された SPARQL Endpoint が設置されている URL を把握する必要があるが，そのどちらも知らないユーザがそれらを知ることは容易ではない．

ユーザが既存の Linked Data リポジトリ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．このようなポータルサイトを，本研究ではデータカタログサイトと呼称する．このデータカタログサイトとしては the Datahub [5]や，Data for Japan [6]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 [6]というオープンソースのデータカタログサイト構築用ソフトウェアを使用しており，登録されたオープン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．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．

ただし，これらデータカタログサイトを利用してユーザが Linked Data リポジトリを発見するためには，リポジトリの公開者が自身で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り，これを怠るとユーザはリポジトリおよび SPARQL Endpoint にアクセスできなくなってしまう．実際に the Datahub には，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，数多く登録されている．

また，データカタログサイトは世界中に複数存在しており，CKAN を使用し構築されたサイトの数は，CKAN を開発している The Open Knowledge Foundation が把握している [7]だけでも 116 件を超えている．このような状況ではユーザにとって，ある Linked Data リポジトリの公開者がどのデータカタログサイトに情報を登録しているかは自明ではなく，

このこともまたユーザにとって未知のリポジトリを発見することを困難にしている．

データカタログサイトを用いない発見としては，リポジトリを公開しているサイト上において，該当するリポジトリに関連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，そもそも要求に合致する Linked Data と，それを公開しているリポジトリの存在をユーザが発見することが困難である．また，例えユーザが何らかの情報源により発見できるとしても，リポジトリが公開されているサイトを発見し，サイト内からリポジトリ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とされるため，ユーザにとって負担となってしまう．

2.4 関連研究

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 [8]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．

また 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 [9]．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 [10]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．

また，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 [11]．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.

Web 上で公開されている Linked Data をクローリングするためのツールとして，LDSpider [13]がある. Tim Berners-Lee が提唱した Linked Data の基本原則の第 3 原則には，“URI を参照したときには標準の技術（RDF や SPARQL 等）を使用して関係する有用な情報を利用できるようにすること” [14]とある．これは即ち，Linked Data 中の URI にアクセスすると，その URI 自身に関する RDF データを返す，もしくはその URI に関する情報を問い合わせる SPARQL 文を RDF データベースに送信した結果を返すようにするべきといった意味であるが，LDSpider はこの原則を利用した Linked Data の自動収集を行っている．探索元となる URI を LDSpider に与えると，LDSpider はその URI にアクセスし RDF データを取得し, さらにその URI から別の URI へと貼られたリンクを辿ることで, 次々と URI 及び RDF データを収集していく. この LDSpider の実際の活用事例としては, State of the LOD Cloud 2014 [14]において 56 万件の URI を探索元とした Linked Data の自動収集に使われている.

この LDSpider による自動収集の欠点としては，探索元として与える URI によって探索範囲が決定してしまうことが挙げられる．Linked Data のリンク関係においては，ある 2 つの URI A と URI B との間に常に双方向のリンクが貼られているとは限らない. 利便性の面では双方向のリンクの方が優れているものの，データの意味としては一方向のリンクでも十分表現できるため，A から B へのみ貼られた一方向的なリンク関係によって繋がっているケースが Linked Data においては多く存在する．そしてこのような場合，URI B を参照しても B から A へのリンクは存在しないため，URI B を探索元として LDSpider を使用した場合には URI A を収集することができない.

この欠点は，Web 上で新しく公開される Linked Data を持続的に収集し続けたい場合において, 特に問題になるものと考えられる. 一般的に考えて, 新たに公開される Linked Data から既存の Linked Data へとリンクを貼ることは比較的容易だが，その逆を行うことには多くの障害がある. まず既存の Linked Data 側にとっては, 新たに公開された Linked Data の存在を知ることが困難である．次に，新たに公開された Linked Data が自らの Linked Data とリンクし得るかどうかを判断することが難しい．更には仮にリンクし得ると判断できたとしても，自分の Linked Data 中のどの URI が，相手の Linked Data 中のどの URI とどのようなプロパティによってリンクするのかを判断することには大きなコストがかかる．これらの障害から，新たに公開された Linked Data に対し既存の Linked Data からのリンクが貼られる可能性は決して高くなく，また貼られるとしてもその作業には長い時間が必要とされる．従って LDSpider は，新たに公開される Linked Data を収集することに不向きであると考えられる.

なおこの問題は、通常の Web ページとそれを収集する Web クローラの場合においても起こり得ることではある．しかし Linked Data のリンクにはリンク元のデータとリンク先のデータ，およびリンクに使うプロパティの種類においてデータの意味としての整合性が求められるのに対し，Web ページにおいてリンクを貼る場合にはその整合性がそれほど求め

られないため，難易度と必要とするコストが低く容易にリンクを貼ることができる．加えて，新たに公開された Web ページへのユーザの誘導は，一般に既存の Web サイトや Web ページから新しい Web ページへのリンクを貼ることで行われる場合が多く，そのリンクを辿ることで Web クローラが収集できる可能性が高いため，Linked Data を収集する LDSpider の場合ほどの問題とはならないものと考えられる．

3. Linked Data リポジトリの自動収集

3.1 Linked Data リポジトリの自動収集の必要性

2.3 節において述べたように, 現在 Linked Data リポジトリの発見に最も利用されているデータカタログサイトは，リポジトリの収集をリポジトリ公開者の自発的な登録に依存しており，それを原因として様々な問題が起こっている．これらの問題は現在のところは人手の作業による対処が可能な規模に収まっているが，セマンティック・ウェブにおける重要技術の一つとしても位置付けられている Linked Data と SPARQL Endpoint はこれからも増加するものと考えられるため，現状のデータカタログサイトによる発見支援には限界が訪れることが予測される．

同様の問題は過去に, 増加し続ける Web ページとユーザが目的の Web ページを発見する際にも生じていた．Web の普及当初に有力であったのは，人手を用いて Web ページを収集し構築するディレクトリ型の発見システムであり，1994 年に開発された当時の Yahoo が特に有名であった．しかし人手の作業では世界中で増加し続ける Web ページに対応できなくなってしまい，そこで 1998 年に登場した Google に代表される，自動的に Web ページを収集するロボット型の検索エンジンを用いる発見支援が主流となった．このように Linked Data リポジトリにおいても，人手の処理に大きく依存している現状のデータカタログサイトに対して，自動的に収集しユーザが検索などを行うことができるようにするリポジトリ発見支援が必要である．

自動収集によるリポジトリ発見支援による具体的なメリットとしては次の三点が挙げられる．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ユーザは変更後の SPARQL Endpoint を捉えられる．第三に，機械的な発見手法を用いることによって，リポジトリ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，データ公開者はユーザからの発見性をそれほど気にすることなく公開できるようになる．

3.2 Linked Data リポジトリに紐づいた Web UI の利用

3.2.1 クローラ型検索エンジンによる WebUI の自動取得

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 などがこの方式を用いている．一般的に，Linked Data リポジトリ公開者によって Web 上のある URL に設置された SPARQL Endpoint は, 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

ーラは SPARQL Endpoint の Web UI も収集していると考えられる

これらの事から，既存のクローラ型検索エンジンを利用することで，その時点において世界中で公開されている Linked Data リポジトリの Web UI を収集することが可能であると考えられる．

3.2.2 特徴的なフレーズを利用した Web UI の抽出

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．図 2 は DBpedia（http://dbpedia.org/sparql）と Linked Geo Data（http://linkedgeodata.org/sparql）にそれぞれ用意された Web UI であり，構築にはどちらも Virtuoso が利用されている．

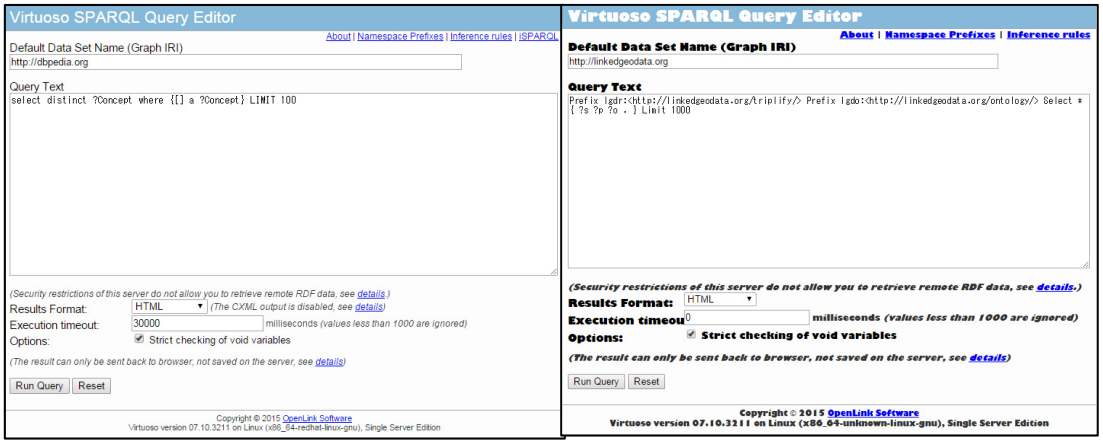

図 2　DBpedia（左）と Linked Geo Data（右）に用意された WebUI

この 2 つの Web UI においては，”Virtuoso SPARQL Query Editor”や”Default Data Set Name（Graph IRI）”などといったように共通して出現する文字列が存在しており，これらは他の Virtuoso を利用した Web UI においても同様に出現するものと考えられる．このような特徴的な文字列を抽出し，これを検索クエリとして Web 検索エンジンで検索することによって，類似した Web UI を持つ SPARQL Endpoint を，公開者の自発的な登録に依存せずに発見できると考えられる．

3.3 収集した Linked Data リポジトリの検索

3.3.1 Linked Data リポジトリの検索における課題

一般的にロボット型検索エンジンを利用した場合の Web ページの検索には，その Web ページ中に出現する単語が手がかりとして用いられることが多い．しかしながら Linked Data という RDF データを扱うための Web サイトである Linked Data リポジトリは一種の Hidden Web であるため，その内容に対する検索は困難である．

まず Linked Data リポジトリが持つ Web UI は，ソフトウェアによる自動生成で構築された汎用的なものである場合が多く，Linked Data リポジトリとしてどのような Linked

Data を扱っているか，などといった情報は記述されていないことが殆どである．加えて他の Linked Data リポジトリと類似することが多いという性質から，出現単語を利用した一般的な索引付け手法では他の Linked Data リポジトリとの違いが表現できない場合が多く，ユーザにとって効果的な検索を提供することができない．

次に，Linked Data リポジトリにおける主要なコンテンツはそのリポジトリにおいて扱われている Linked Data であると考えられるが，これを効果的にサンプリングすることは難しい．

たとえばある Linked Data リポジトリが持っている RDF データの中から，単語の出現頻度解析などを目的として文字列データを抽出することを考えるとき，SPARQL における RAND()演算子を利用し

```
SELECT * WHERE  { ?s ?p ?o.
                  FILTER( isLiteral(?o) )
                  } ORDER BY RAND() LIMIT 10
```

といった SPARQL 式を書いて SPARQL Endpoint に送信すると，その Linked Data リポジトリが扱っている RDF データの中から，目的語がリテラル（文字列）であるトリプルをランダムで 10 件取り出すことができる．このような SPARQL 式を活用することで Linked Data リポジトリで扱われている Linked Data のサンプリングを行うことそのものは，必ずしも不可能とは言い切れない．

しかしながら上記の SPARQL 式を Virtuoso に対して送信した場合，エラーこそ起こらずに結果が返されるものの，結果にアルファベット順の並びが見られるほか，数回の試行において同じ結果が返されるなどといった挙動が見られ，想定通りのランダムな抽出ができているとは言い難い．更に RAND()は SPARQL の最新版である SPARQL 1.1 の構文であるため，全ての SPARQL Endpoint が SPARQL 1.1 に対応しているとは限らない現状において一律に適用することには問題がある．また，RDF によって記述される Linked Data の内容は非常に多彩であり，ある Linked Data リポジトリにおいてどのようなデータが独自性のあるデータであるか，どのような条件のサンプリングであれば検索に都合の良いデータが取得できるかという問題が残る．加えて，Linked Data リポジトリに格納されているのが必ずしも公開者独自のデータのみであるとも限らず，利用上の都合のために一般的な RDF データを一緒に格納している場合も多いため，SPARQL Endpoint を介した Linked Data リポジトリからの適切なサンプリングは，容易に解決できる問題ではないと考えられる．

また，サンプリングではなく全データを取り扱うことを考えた場合にも問題が残る．SPARQL Endpoint はあくまでも HTTP リクエストとそれに対する HTTP レスポンスの形で処理を行う Web API であり，膨大なデータ量を一度に扱うことをそれほど得意としていない．加えて SPARQL Endpoint が使用している SPARQL エンジンやそれを動かしている実際の Web サーバが，現状ではそれほど強力ではないといった問題もあり，膨大な量のデータを取得するような SPARQL 式を送信した場合にはサーバ側で設定されたタイムアウト

時間を容易に超過し，データの取得が中止されてしまうのである

3.3.2　関連ドキュメントの取得と利用

前節で述べたように Linked Data リポジトリの Web UI は自らについての記述を持たない場合が多く，これを公開するだけでは利用者にはそれがどのような Linked Data リポジトリであるか全く分からない．そのため Linked Data リポジトリの公開者は，自らが公開するリポジトリの内容について説明するようなドキュメントを，何らかの形で公開しているものと考えられる．以降，本研究ではこのようなドキュメントを“関連ドキュメント”と呼称する．

ここで，http://www.opmw.org/sparql という URL に設置されている SPARQL Endpoint を例に挙げて説明する．この SPARQL Endpoint は Virtuoso を利用して構築されており，この URL にアクセスしても図 3 左のような Web UI しか表示されず，利用者にはこの Linked Data リポジトリでどのような RDF データが提供されているのか見当が付かない．しかしこの Web UI を説明するようなドキュメントが図 3 右のような Web ページとして用意されており，これを読むことによって利用者はこの Linked Data リポジトリで扱われている RDF データが，Towards Workflow Ecosystems Through Standard Representations という研究の成果物であるなどといった情報を得ることができる．

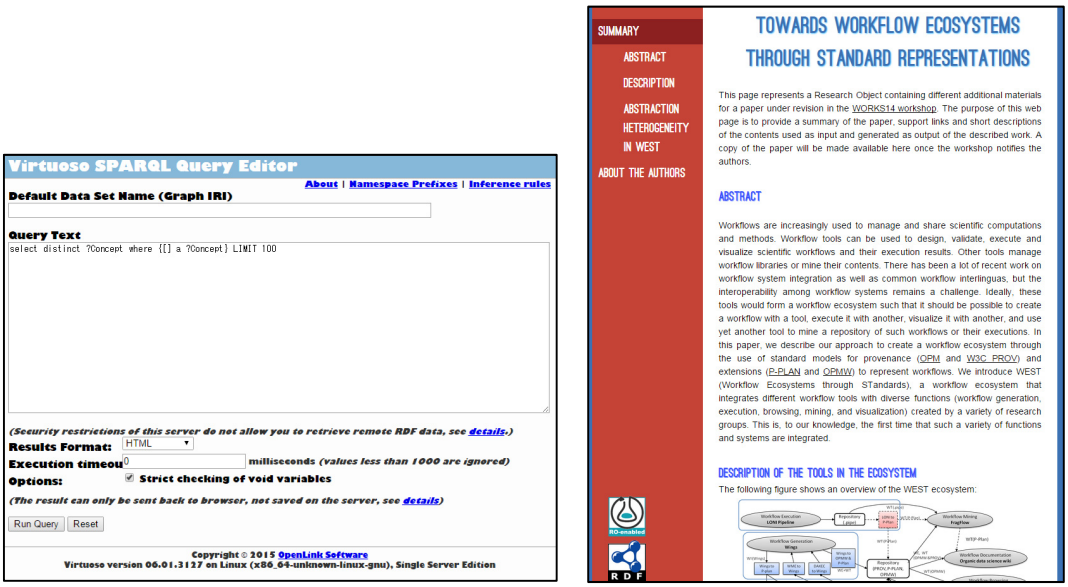

図 3　Web UI（http://opmw.org/sparql）と，それを説明する関連ドキュメント（http://rohub.linkeddata.es/WORKS14-WfEcosystems/index.html）

このように関連ドキュメントは，利用者に読ませることで Linked Data リポジトリに関する情報を与えることを目的として書かれた Web ページであり，従って対応するリポジトリを表現するような単語などといった情報が含まれている．この事から，この関連ドキュ

メントを利用することによって Linked Data リポジトリの効果的な検索が実現できると考えられる．

また，同じく Web ページである Web UI に対して文書中でハイパーリンクを張る，もしくは Web UI の URL を記述するなどの手段によって，利用者を Linked Data リポジトリへと誘導する役割を持たされていることが自然である．このリポジトリへの誘導を利用することで，クローラ型検索エンジンを利用し自動収集したリポジトリを手がかりとして，そのリポジトリ自身に対する関連ドキュメントを自動収集することが可能であり，リポジトリの自動収集からユーザへの検索機能提供までをほぼ機械的に行うことができると考えられる．

4. クローラ型検索エンジンを利用した自動収集手法の実現

4.1 Web UI のサンプル取得

Web UI に特徴的な文字列を抽出するためには，Web UI のサンプルを用意する必要がある．現在 SPARQL Endpoint の収集が最も容易であるのは，the Datahub を利用することである．

データカタログサイトである the Datahub には LOD が多く登録されているが，同時に LOD ではないオープンデータや，ダウンロードによる利用を前提とし SPARQL Endpoint を持たない LOD もまた多く登録されている．そのため，the Datahub の機能として各データセットのメタデータとして付与された検索用タグを利用する．図 4 は the Datahub において，DBpedia というデータセットに対し様々な検索用タグが付与されていることを示す画面である．

図 4　the Datahub におけるデータセットと検索用タグ

the Datahub では Linked Data リポジトリ公開者が登録したデータセットに対する利用者の検索を円滑にするため様々な検索用タグが付与されるが，データセットを提供するためのフォーマットとして SPARQL Endpoint が用意され登録されていることを示すためのタグとして，”api/sparql”タグが存在する．CKAN API を通じて the Datahub に検索クエリを送信し，”api/sparql”タグを付与されているデータセットを取得することで，SPARQL Endpoint である URL を収集することができる．

ただし，the Datahub に登録されているデータには既に URL へのアクセスができなくなっているデータも多く存在し，また有効ではあっても Web UI を持たない SPARQL Endpoint も存在する．従って取得した URL に対しては，取得時点で未だ有効であるか否か，Web UI を持っているか否かの 2 点について調査する必要がある．

4.2 Web UI サンプルのクラスタリング

Web UI を収集する目的は共通して出現する文字列を抽出するためであるが，この共通文字列は構築に利用されたソフトウェアや，SPARQL Endpoint 構築のプロジェクトなどといった背景から，構築された Web UI が類似することによって出現する．従って抽出すべき共通文字列は Web UI 全てに共通して現れることはなく，基本的には幾つかの類似 Web UI 群に着目した時に発見することができる．このことから，前項で取得した Web UI のサンプル群について，Web ページとしての類似性に基づいた，しかし非階層的なグループ分けを

行うクラスタリングによって，類似 Web UI 群ごとに分類する必要がある．

このような目的のクラスタリングについて，本研究では MDS（MultiDimensional Scaling：多次元尺度構成法）を使用した方法を取る．MDS は，“空間内の距離によって，変数間の関係の強弱を表そうとする多変量解析”であり，”変数間の類似性をもとにして，変数を最小次元数の空間内に付置することをめざす”方法である [16]．対象である複数の変数に対して，全ての変数間の類似性，つまり距離として扱う類似度の数値をインプットとして与えると，アウトプットは 2 次元や 3 次元といった人が認識しやすい低次元空間上に，距離が近い変数は近い位置に，距離が遠い変数は遠い位置にプロットされた点の集合として出力される．すなわち，Web UI を変数，Web UI 間の類似度を距離として MDS を利用すれば，類似した Web UI ごとにある位置に集められてベクトル空間にプロットされた，各 Web UI に対応する点ベクトルの集合が取得できる．このベクトルの集合を，非階層的クラスタリング手法の一つである K-means（K 平均法）を使用してグループ分けを行い，これを類似 Web UI 群として取得する．

ある 2 つの Web ページについて類似性を測る際の尺度には大別して，Content Similarity と Structural Similarity の 2 種類が存在する．Content Similarity とは対象とする Web ページ中に出現する Content に着目した場合の類似度であり，一般的には Web ページ中に出現する単語や文字列がどれだけ類似しているかを扱う．対して Structural Similarity は Web ページを HTML 文書として見た時の文書構造の類似度であり，主に Web ページとしての構造がどれだけ類似しているかを扱うものである．

Web ページ間の Content Similarity の計算方式はいくつか存在する．単語を次元としたベクトル空間モデルによって各々の Web ページを表現し，Web ページの単語ベクトル同士の類似度を内積やコサイン類似度を用いて計算する方法や，文字列同士の編集距離（ユークリッド距離）による方法などがある．本研究では編集距離を使用して Content Similarity を計算している．

Structural Similarity についても多くの計算方式が提案されているが，本研究では木編集距離（Tree Edit Distance）を用いて算出する．木編集距離はある 2 つの木について，一方の木が図 5 で図解されているような要素の追加・削除・置換の各編集操作により，もう一方の木と同じ構造になるまでの操作数をもって類似性を測る方法である．

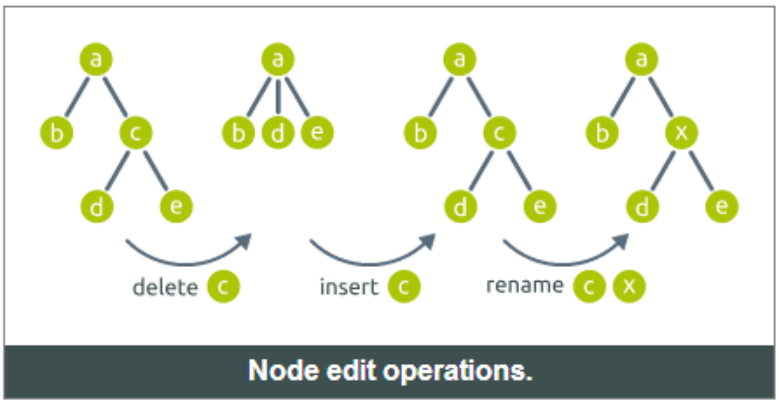

図 5　木編集距離における削除，追加，置換の操作
(http://www.inf.unibz.it/dis/projects/tree-edit-distance/tree-edit-distance.php より)

これら 2 種類の類似性を考慮するための最も容易な手法としては，Content Similarity による類似度と Structural Similarity による類似度のそれぞれに重み付けをした上で足し合わせたものを類似度とするものがある．

$$\mathrm{Sim}(a,b) = \mathrm{ContentSimilarity}(a,b) \times \alpha - \mathrm{StructuralSimilarity}(a,b) \times (1-\alpha)$$

しかしながら本研究で目的としているのは，あくまでも類似する Web UI ごとのクラスタリングである．Web UI の類似の理由の第一は同じ構築ソフトウェアによる自動構築だが，この際 Web UI 構築者の必要に応じて，ソフトウェアが自動構築した Web UI にカスタマイズが加えられることがある．この際のカスタマイズがどの程度まで及ぶかは，使用ソフトウェアと構築者の要求に依存するが，一般に Web ページの編集においては，HTML 構造をそのままに構造内の文章のみを変更する方が容易であり，また HTML 構造まで踏み込むようなカスタマイズの場合にも，元の構造をある程度流用する可能性が高い．従って，本研究の類似 Web UI 群の判別の用途においては，Structural Similarity の方がより有効であると考えられる．

しかしながら，Structural Similarity のみで目的が達成できるわけではない．Web UI である Web ページにおいては，SPARQL クエリを入力するためのテキストフォームと送信ボタン等といった必要要素がそもそも少ないため，異なるソフトウェアを使用し構築された Web UI であってもシンプルなデザインに設計されている場合には，木編集距離を利用した Structural Similarity のみでは十分に距離のある類似性を表現できず，それを利用するグループ分けが困難である．かといって先述した Content Similarity による類似性との合成を安易に採用しては，Content Similarity の影響が Structural Similarity で適切にグループ分けができるケースにまで及んでしまう危険性がある．

そこで本研究では Structural Similarity を用いた分類を行った後，その分類結果の各クラスタに対して Content Similarity による分類を行う 2 段階の分類によって，Structural Similarity による分類能力と，Content Similarity による細かい分類とを両立している．図

6 はこの 2 段階の分類アルゴリズムを擬似コードで表したものである．

```
WebUIClustering(SampleWebUIs) # SampleWebUIs are all samples of Web UI
        X = StructuralSimilarityClustering(SampleWebUIs)
        result = Array.new
        for each x in X              # X is a set of StructuralSimilarity Clusters
                if x.size > n        #n is a threshold
                        Y = ContentSimilarityClustering(x)
                        for each y in Y
                                result << y
                        end
                else
                        result << x
                end
        end
        return result
end
```

図 6　Structural Similarity と Content Similarity を利用した 2 段階分類

4.3　n-gram に基づく類似 Web UI 群からの特徴フレーズ抽出

4.2 節で述べた手法によって分類し取得した類似 Web UI 群について，それらの中において出現する特徴的なフレーズを抽出する．この段階において考慮すべき点として，この作業は 4.4 節で行う既存のクローラ型検索エンジンで使用するためのフレーズを取得するためのものであるということがある．現在稼働している有名な検索エンジンには様々なチューニングが施されているが，その中の一つに検索エンジンが収集した Web ページの内容に対し，単語辞書を用いた単語単位でのインデックスを張る技術が広く使用されている．そのため本研究で抽出する文字列は，単語の並びとして成立するような文字列である必要がある．

そこで本研究では，単語単位の n-gram を利用して連続した単語からなるフレーズを抽出し，それらフレーズについて類似 Web UI 群における出現頻度をカウントする．この単語単位の n-gram について n=3 の場合を例にすると，図 7 のような 3 語からなるフレーズごとのカウントとなる．

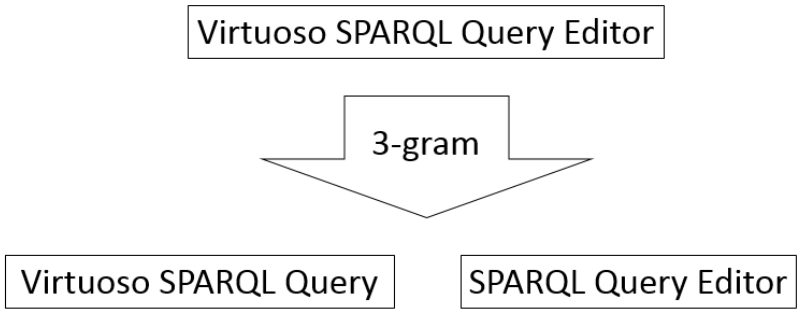

図 7　単語単位の n-gram の例

一般にこの n の値が大きいほど，よりフレーズとしての特定性が高く有用である．しかし n が大きすぎると逆に類似 Web UI 群中での出現数が少なくなる可能性もあるため，本研究では n=5 を採用した 5-gram によって出現頻度を取り，類似 Web UI 群中において頻度が高かったフレーズを抽出する．

なお，Web UI である Web ページにおいては，SPARQL クエリ入力のためのテキスト入力フォームの初期値をフレーズ抽出の対象から除外している．SPARQL クエリ記述においては使用する URI の接頭辞を PREFIX 句で指定しそれ以降の SPARQL クエリの一部を省略する工夫がされることが多いが, この PREFIX 句を Web UI ではユーザの SPARQL クエリ入力支援のために入力フォームの初期値として予め記述しておく場合がある．従ってこの PREFIX 句が含まれるフレーズは Web UI において頻出することが予想されるが，これは同時に SPARQL クエリ一般においても頻出するフレーズであるため，4.4 節で扱う検索サービスを利用した Web UI の取得におけるノイズとなってしまうので除外した．

4.4　既存の検索サービスを利用した SPARQL Endpoint の取得

4.4.1　既存の検索サービス選定とその利用方法

4.3 節で取得したフレーズをクエリとして，既存の検索サービスを利用した検索を行い，SPARQL Endpoint の Web UI である Web ページを取得する．この際に利用する検索サービスの条件としては，第一に“指定した複数の単語が指定通りに並んで出現する”ことを条件とした検索ができるフレーズ検索の機能を持っていること．第二に，その検索サービスがコピーコンテンツや類似コンテンツへの対策機能を備えている場合，それをユーザ側からの設定によってある程度制御できることが必要である．近年，Web ページの内容を他の Web ページからの過剰な引用もしくは複製によって構成するコピーコンテンツが流行しており, これらは Web ページ検索においては非常に類似した Web ページばかりを提示してしまう事態を引き起こし問題となる．そのため現在の検索サービスにおいては，他の Web ページとの類似度が高い Web ページをコピーコンテンツと見なしランキングを下げる，またはユーザへ提示する検索結果から除外するといった対策をとっている場合がある．本研究の手法において着目している類似 Web UI 群は，Web ページ単体として見た場合はコピーコンテンツと見なされる可能性が高いため, 上記のような対策機能は Web UI 取得のため

の障害となってしまうのである．

上記の条件を満たす既存の検索サービスとして，Google 検索(http://www.google.co.jp)がある．Google 検索では検索クエリをダブルクォーテーションで囲むことでフレーズ検索を行うことができる．また Google 検索はコピーコンテンツの対策として類似 Web ページを検索結果から除外する機能を備えているが，filter=0 のパラメータ設定を送信する URL において付与することによって，この除外機能を停止させて利用することが可能である．実例を示すと，“strict checking of void variables”というフレーズ検索によって Web UI を取得したい場合，以下のような URL によって検索を行う．

http://www.google.co.jp/search?q=”strict+checking+of+void+variables”&filter=0

4.4.2　SPARQL Endpoint の判定

4.4.1 節で述べたフレーズ検索によって取得する検索結果の中には，Web UI ではない Web ページが含まれている可能性が常に存在する．従ってフレーズ検索により取得してきた Web ページ群について，それが SPARQL Endpoint の URL であるか，または SPARQL Endpoint に紐付いた Web UI であるかを判定する必要がある．

まず, SPARQL Endpoint であるかどうかの判定については, 検索によって取得した URL を SPARQL Endpoint と見なした上で, 単純な SPARQL 式を送信することによって判別が可能である．たとえば，次のような SPARQL 式を考えてみる．

```
ASK WHERE { ?s ?p ?o }
```

これは SPARQL のクエリ形式の中の一つである ASK 形式を利用した SPARQL 式であり, WHERE 句以降に記述されたブロックパターンが対象の RDF データの中に存在しているか否かを返させるためのクエリ形式である. SPARQL Endpoint はブロックパターンが存在していた場合には true を，存在しない場合には false を返してくる．上記の SPARQL 式はブロックパターンを全て変数にしており，これはどのようなトリプルであってもマッチするようなパターン条件となっている．従って，ある URL にこの SPARQL 式を送信した際の反応は，(A)true が返される，(B)false が返される，(C)エラーが返される，の 3 種類である．この中で B の場合は SPARQL Endpoint として機能はしているが DB には RDF データが全く格納されていない，といったケースであると考えられるため，Linked Data リポジトリの SPARQL Endpoint であると判定するべきは A の場合のみである．

しかしながら稀に ASK 形式の SPARQL 式をサポートしていない SPARQL Endpoint も存在しており，The British National Bibliography が公開している Linked Data リポジトリの SPARQL Endpoint（http://bnb.data.bl.uk/sparql）などはそれに該当している．このようなケースへの対応のため，SPARQL Endpoint の利用においてより普遍的に使われる SELECT 形式を用いた SPARQL 式による判別方法を考える必要がある．

```
SELECT * WHERE { ?s ?p ?o } LIMIT 1
```

この SPARQL 式を送信した際の反応は，(A)なんらかのトリプルが 1 件返される，(B)トリプルが 1 件も返されないがエラーも返されない，(C)エラーが返ってくる，の 3 種類であ

り，ここで A に該当する反応を返した URL については SPARQL Endpoint であると判定することができる．

4.4.3 Web UI からの SPARQL Endpoint 抽出

次に, SPARQL Endpoint の URL ではないが SPARQL Endpoint に紐付いている Web UI について述べる．Web UI とは SPARQL 式の入力と SPARQL Endpoint への送信の手続きをブラウザ上で行うための Web ページであり，それが設置される URL は利便性や構築ソフトウェアの仕様から SPARQL Endpoint の URL と一致させていることが多い．しかしながら Linked Data リポジトリの構築と Web UI の構築とをそれぞれ別のツールによって行い，それぞれを別の URL に設置した上で，Web UI から SPARQL Endpoint の URL へと送信しているケースが存在する．例として“STW Thesaurus for Economics”の Linked Data リポジトリでは, SPARQL Endpoint の URL（http://zbw.eu/beta/sparql/stw/query）と Web UI の URL（http://zbw.eu/beta/sparql/stwv.html）とが異なっている．この例においては Web UI の HTML 構造における form タグ中の action 属性によって SPARQL 式の送信先が記述されており，従って同様のケースについては取得した Web UI 中の action 属性を参照することによって Web UI に紐付いている SPARQL Endpoint を取得することが可能である．

ただし任意の URL に SPARQL クエリを送信すること自体は，HTML の form タグ以外の方法によっても容易に実現できるため，action 属性を参照する本研究の手法のみによって Web UI と SPARQL Endpoint の紐付き方のパターン全てに対応することは困難だと考えられる．この点においては新たに発見されるパターンへの継続的な対応が必要である．

4.5 関連ドキュメントを利用した Linked Data リポジトリ検索

4.4 節の手法で取得できる SPARQL Endpoint の URL を手がかりとして，その Linked Data リポジトリを説明するような関連ドキュメントを取得する．この関連ドキュメント収集における URL の利用法として本研究では，（A）URL に対するバックリンクを調べる，（B）URL を文字列クエリとして検索を行う，（C）URL のドメイン部を URL としてアクセスする，の 3 種類の手法を提案する．

（A）URL に対するバックリンクを調べる

Web UI は Web ページであるため，それを説明する関連ドキュメントである Web ページからリンクを張ることができる. 従って Web UI に対するバックリンクを調べることによって取得できる Web ページの中には，関連ドキュメントが含まれているものと考えられる．

ある URL に対するバックリンクを調べる方法としては，Google 検索において link:演算子を利用したクエリを構築することで Google が自らのクローラで調査したバックリンクを取得することができる．しかしながら，Google が提示するバックリンク先の Web ページは他のサービスを利用した場合と比べて明らかに数が少なく，ランキングアルゴリズムに用

いられているスコアによって何らかの形で閾値が設けられている可能性がある．そのため本研究では，Moz（http://moz.com）が提供している Open Site Explorer（https://moz.com/researchtools/ose/）というサービスを利用する．Moz は専用の Web クローラを稼働することで Web ページに関する情報を収集し，ユーザが要求した URL に対する様々な分析を提供する SEO サービスであるが，そこで提供される情報の中にはバックリンクが含まれている．これは Web UI の収集時に利用する Web クローラとは別のクローラによって収集されたリンク関係の情報であるため, 必ずしも全ての Web UI についてバックリンクが発見できるとは限らないが，しかし Google を利用した場合よりも多くのバックリンクを発見することが期待できる．

（B）URL を文字列クエリとして検索を行う

SPARQL Endpoint は Web API であるため，ユーザが利用するためには URL を把握している必要がある．従ってユーザのために SPARQL Endpoint について記述しているような関連ドキュメントにおいては，少なくともドキュメント中に SPARQL Endpoint の URL を記述している可能性が高いと考えられる. 従って SPARQL Endpoint の URL を文字列とした検索を行うことによって，目的とする SPARQL Endpoint の関連ドキュメントを取得できると考えられる．

（C）URL のドメイン部を URL としてアクセスする

Linked Data リポジトリおよび SPARQL Endpoint を公開者が公開する際は，公開者が運営する Web サイトの一部として URL を用意することが多い．この際，公開している Linked Data リポジトリにおいて扱っている RDF データの内容は，公開者が運営している Web サイトの内容と関連している可能性が高いと考えられるため，Web サイトのタイトルや主題などが記述されている可能性が高いトップページはリポジトリの関連ドキュメントとして使用できる可能性がある．そこでトップページにアクセスするために，SPARQL Endpoint の URL のドメイン部を URL としてアクセスする. 例として DBpedia の SPARQL Endpoint（http://dbpedia.org/sparql）の場合には，“http://dbpedia.org/”にアクセスして Web ページを取得する．

以上の 3 種類の手法によって収集した関連ドキュメントを利用して，SPARQL Endpoint の URL に対するユーザの検索を支援する方法を考える．単純な発想としては，取得した関連ドキュメントから Linked Data リポジトリに関する記述の部分を抽出し，SPARQL Endpoint を検索するためのメタデータとして付与したうえで検索システムを構築するような方向性が考えられるが，ここには多くの困難が存在する．

関連ドキュメントは Linked Data リポジトリ運用の必要条件というわけではなく，関連ドキュメント記述のためのフォーマットなども特に定められていないため，関連ドキュメントは Linked Data リポジトリ公開者それぞれのスタイルによって記述される．このことから，関連ドキュメントにはメタデータ付与に必要とする情報全てが常に記述されるとは限らず，記述された関連ドキュメントのどの部分に Linked Data リポジトリの情報である

かは自明でなく，関連ドキュメントは基本的にユーザが人手で読むことを想定したドキュメントであるためその内容を解釈するような機械的処理が困難であるといった問題がある．

以上の問題の存在から，本研究では関連ドキュメントからメタデータを抽出すること以外の方向による SPARQL Endpoint の検索の実現を目指す．

SPARQL Endpoint に対する検索が困難であることは今までに述べている通りであるが，その関連ドキュメントに関してはあくまでも Web ページ，もしくは手法 B で利用した検索エンジンによって取得できる PDF 等であり，こちらに対する検索は既に確立された技術であり容易に実現可能である．そこで本研究の SPARQL Endpoint に対する検索は，関連ドキュメントに対する既存の全文検索システムと，関連ドキュメントの取得段階で構築した SPARQL Endpoint と関連ドキュメントの紐付きによって実現させる．図 8 は本研究で実現させる Linked Data リポジトリ検索システムとユーザの関係を図で表したものである．

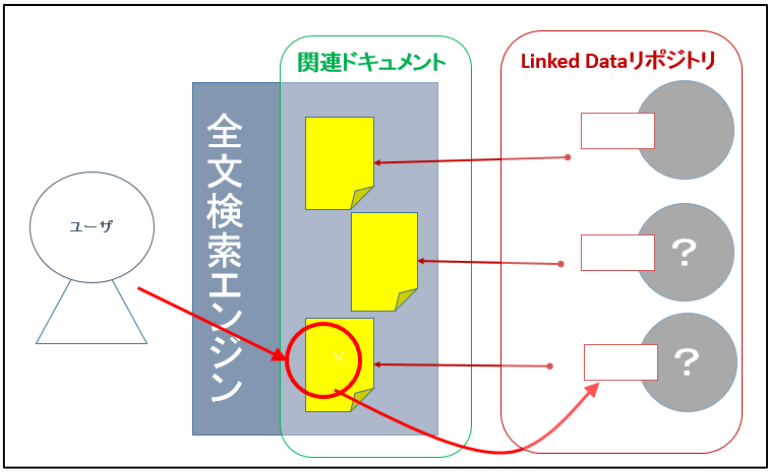

図 8　Linked Data リポジトリ検索システムの模式図

4.4 節において取得した SPARQL Endpoint に対して，本節前半で述べた手法によって取得した関連ドキュメントを全文検索エンジンに格納し，ユーザにはまず関連ドキュメントに対する検索機能を提供する．そして検索によってユーザが発見した関連ドキュメントについて，それに紐付いた Linked Data リポジトリを提示することによって，ユーザの Linked Data リポジトリ発見を支援する検索システムとするものである．

本研究における検索手法の利点として，Linked Data リポジトリの関連ドキュメントの集合を文書集合とした検索が行えることがある．一般的な Web 検索エンジンで同等の検索を行うためには，検索集合を Linked Data リポジトリと関連する Web ページに限定するような検索語と併せた検索を行う必要があるが，ユーザがそのような検索語を考案するためには Linked Data リポジトリに一般的な傾向について熟知している必要があり，加えてその検索語によってユーザが検索したいキーワードの表現能力が干渉される恐れがある．この点において本研究の検索システムは，文書集合そのものが Linked Data リポジトリの関連ドキュメントのみに限定されているため，ユーザが検索したいキーワードに影響を及ぼすことなく検索させることができ，また発見したドキュメントに対して確実に Linked Data

リポジトリを提供することが可能である.

またデータカタログサイトと比較した場合の利点として，Linked Data リポジトリ公開者の自発的な登録に依存することなく機械的に収集および更新が可能であることの他に，多くのユーザが慣れ親しんでいる Web 検索エンジンと同様の使用感によって Linked Data リポジトリの発見が行えるようになることが挙げられる.

5. 検証と考察

5.1 Web UI サンプルの取得・類似 Web UI クラスタリング・特徴フレーズ抽出

本研究の根幹であるクローラ型検索エンジンを利用した Linked Data リポジトリの発見能力について，実際に評価実験を行う．

2014 年 11 月 16 日に the Datahub より取得したデータセットには，表 1 のような内訳で Web UI のサンプルが含まれていた．

表 1　2014 年 11 月 16 日に the Datahub から取得したサンプルの内訳

取得データセット情報	有効 SPARQL Endpoint 数	Web UI 数
499	312	254

この Web UI サンプルを 4.2 節の手法に基づいてクラスタリングを行ったところ，5 件以上の Web UI を含むクラスタは 10 個構成された．本実験はその中で，人手による正解 Web UI の判別が容易であり正確に行えた 3 個のクラスタを用いて検証する．

この 3 個のクラスタはそれぞれ，主に Virtuoso を使用した Web UI によって構成されるクラスタ，SPARQLer [17]を使用した Web UI によるクラスタ，The RKB Explorer [18]を利用した Web UI によるクラスタであり，以降はこれらを”Virtuoso クラスタ”，”SPARQLer クラスタ”，“RKB クラスタ”と呼称する．これらクラスタにおける構成 Web UI 数と，その中で正しく分類されたと判断できる正解 Web UI 数を表 2 に示す．

表 2　3 個のクラスタにおける Web UI の内訳

	構成 Web UI 数	正解 Web UI 数
Virtuoso クラスタ	101	101
SPARQLer クラスタ	12	11
RKB クラスタ	48	48

表 2 において構成 Web UI 数と正解 Web UI 数がほぼ同じ値となっていることから，少なくとも今回注目する 3 個のクラスタそれぞれについては，正しく類似 Web UI を集めたクラスタリングが実現できていると言える．

次に表 2 で示した 3 個の類似 Web UI クラスタについて, 4.3 節で述べた手法を用いて特徴的なフレーズを抽出する．この処理によって抽出されるフレーズは多数存在するが，その中で各クラスタごとに最も頻度が高かったフレーズの中の 1 つを例として，4.4 節で述べたクローラ型検索エンジンによる SPARQL Endpoint 取得の実験を行う．なお 5.3 節において後述するが，検索エンジンから取得できる SPARQL Endpoint の内容は使用するフレーズごとに少しずつ異なる. そのため 1 種類のフレーズのみを使用して SPARQL Endpoint の取得を行う場合には，フレーズの選び方によって取得 SPARQL Endpoint 数で見た収集性能に差が生じる．だが本研究では複数のフレーズを併用することで 1 種類のみを使用した場合よりも向上させられることを示したため，この段階で選ぶフレーズはあくまでも検

証のための一例として，同じ出現頻度のフレーズの中から収集性能が極端に悪くなかったフレーズを一つ選んでいる．この観点から抜粋したフレーズを，表 3 において各クラスタ中における出現頻度と併せて示す．

表 3　実験に使用する特徴的フレーズ

	特徴的フレーズ	出現頻度
Virtuoso クラスタ	“back to browser not saved”	94
SPARQLer クラスタ	“xslt style sheet blank for”	11
RKB クラスタ	“above form is currently configured”	48

表 3 に示した 3 種の特徴的フレーズのいずれにおける出現頻度も，表 2 で示した構成 Web UI 数の 9 割以上に達している．これはすなわち各類似 Web UI クラスタにおける 9 割以上の Web UI に共通して出現するフレーズを抽出できているということであり，期待した通りの特徴的フレーズ抽出が実現できていると言える．

5.2　提案手法における SPARQL Endpoint 収集能力

本研究が提案する Linked Data リポジトリの自動収集手法の SPARQL Endpoint 取得能力について，表 3 に挙げたフレーズを用いて 2 つの観点から評価する．

まず網羅性の観点から，検索エンジンから取得した SPARQL Endpoint によって，the Datahub に登録されている既知の SPARQL Endpoint をどの程度再現できるかを見る．表 4 は各クラスタの特徴的フレーズを用いて取得できた the Datahub に登録のある SPARQL Endpoint 数と，the Datahub に登録のある SPARQL Endpoint の中で各クラスタの分類に該当する SPARQL Endpoint 数，及び取得 Endpoint 数を検索ドキュメント数とし該当 Endpoint 数を全適合ドキュメント数としたときの再現率を表したものである．

表 4　既知の SPARQL Endpoint に対する再現率（無効 Endpoint を含む）

	取得により再現できた Endpoint 数	the Datahub 中の Endpoint 数	再現率
Virtuoso クラスタ	21	111	0.19
SPARQLer クラスタ	12	15	0.80
RKB クラスタ	34	48	0.71

表 4 の結果からは SPARQLer クラスタと RKB クラスタで 0.7 以上と高い再現率を出している一方で，the Datahub 中の件数が多い Virtuoso クラスタにおいては 0.2 未満と低い値を示している．

次に the Datahub に登録されていない，未知の SPARQL Endpoint に関する収集能力を調べる．表 5 は，各クラスタの特徴的フレーズを用いた Google 検索でのフレーズ検索によって取得できた SPARQL Endpoint 数と，その中に含まれていた the Datahub 未登録の

SPARQL Endpoint 数を表したものである.

表 5　取得 SPARQL Endpoint 数と，the Datahub 未登録 Endpoint 数の内訳

	取得 SPARQL Endpoint 数	the Datahub 未登録 SPARQL Endpoint 数
Virtuoso クラスタ	59	38
SPARQLer クラスタ	22	10
RKB クラスタ	41	7

表 5 の結果から，Virtuoso クラスタの特徴的フレーズを利用した場合に，the Datahub 未登録の SPARQL Endpoint を最も多く取得できていることがわかる．このことから表 4 における Virtuoso クラスタの再現率の低さについては，Virtuoso クラスタに該当する Web UI について本研究の手法が機能しない，といった問題であるとは考え難い.

ここで Virtuoso クラスタに属し，the Datahub に登録がされており，かつ本研究の発見手法において収集できていない SPARQL Endpoint の例として，生命情報学に関する Linked Data を提供する Bio2RDF プロジェクトの SPARQL Endpoint がある. 生命情報学はオントロジー，すなわち RDF データの利用が盛んな学術分野であり，多くの研究成果が Web 上のデータベースとして公開されている. Bio2RDF はこれらのデータベースを集積し，生命情報学の統合データベースとしてユーザに提供することを目指したプロジェクトである．この Bio2RDF をユーザが利用する方法として，データセットごとに Virtuoso を利用し構築した SPARQL Endpoint が用意され，Bio2RDF のトップページ(http://bio2rdf.org/)からデータセットそれぞれの SPARQL Endpoint へとアクセスできるようになっている．本研究で the Datahub から取得したサンプルの中には，この Bio2RDF が提供する SPARQL Endpoint が 39 件含まれていたが，表 4 の実験において取得できたのはその内の 2 件のみであった.

これらの SPARQL Endpoint に対する取得が失敗した理由として，Google 検索の重複コンテンツに対するインデックス登録のルールの影響が考えられる．ここで扱う重複コンテンツとは“ドメイン内または複数ドメインにまたがって存在する，他のコンテンツと完全に同じであるか非常によく似たコンテンツのまとまり”[19]を指しており，たとえばある Web ページについて印刷用バージョンのページが同時に存在する場合などがこの重複コンテンツに該当する．Google では固有の情報を持つページをインデックスに登録するよう努めていると言っており，たとえば上記の印刷用バージョンのようなケースにおいては，どちらか 1 つの Web ページのみを登録するとしている.

本研究でリポジトリ収集の手がかりとする類似 Web UI 群は，すなわち非常によく似た Web ページの集まりとも言えるため，Google 検索から重複コンテンツと見なされる可能性が高いものと考えられる. 今回収集できなかった Bio2RDF の SPARQL Endpoint について調査を行ったところ，確かに the Datahub に登録がされており，the Datahub 上におけるそれぞれのデータセットのページが Google 検索のインデックスに登録されており，更には

そのページから SPARQL Endpoint の URL へとリンクが貼られているにも関わらず，SPARQL Endpoint の URL にはインデックスが作られておらず，人手によるキーワード検索によっても発見することができなかった．このことから，今回の実験で取得できなかった Bio2RDF の SPARQL Endpoint については，Google 検索の重複コンテンツに対するルールの影響によって，提案手法での取得が適わなかったものと考えられる．

ただし，表 4 の実験においては Virtuoso クラスタ以外ではある程度高い再現率を出しており，また表 5 の実験では Virtuoso クラスタについても the Datahub 未登録の SPARQL Endpoint を確かに発見できていることから，少なくとも類似 Web UI が複数ドメインにまたがって存在する場合には重複コンテンツとは見なされていない，もしくは見なされたとしても登録そのものを削除される程の対応はされていないと推測される．よってこの重複コンテンツへのルールによる問題は，Bio2RDF のような同一ドメイン上に複数の SPARQL Endpoint を用意しているケースにおける問題であり，本研究の収集対象全てが影響を受けるわけではないものと考えられる．

また，他に提案手法において取得できない SPARQL Endpoint の例として，Linked Geo Data（http://linkedgeodata.org/sparql）がある．Linked Geo Data は Virtuoso を使用し構築された SPARQL Endpoint であり，the Datahub への登録・更新も行われているが，robots.txt によりクローラの巡回を拒否している．つまり Linked Geo Data は，クローラが収集した Linked Data リポジトリを取得する本研究の手法においてそもそも取得することができない Linked Data リポジトリであったと言える．このような robots.txt の影響により取得できない SPARQL Endpoint は，the Datahub から取得した Virtuoso クラスタの SPARQL Endpoint において 4 件見られており，このこともまた表 4 において Virtuoso クラスタの再現率が低くなった一因と考えられる．

また表 5 における SPARQLer クラスタと RKB クラスタの結果についても，Virtuoso クラスタの取得数にこそ及ばないものの，the Datahub 未登録の SPARQL Endpoint を発見することができている．このことから，本研究の Linked Data リポジトリ発見手法は，サンプルに使用した the Datahub に登録していない Linked Data リポジトリについても，確かに収集することができると言える．

5.3 特徴的フレーズにおけるヒューリスティクス

次に，類似 Web UI クラスタから抽出する特徴的フレーズについて扱う．4.3 節では類似 Web UI クラスタにおいて出現頻度が高いフレーズを採用するとしたが，実際には同じ出現頻度のフレーズが複数出現する場合が多いため，これら複数のフレーズ候補への対応を考える必要がある．

表 6 は Virtuoso クラスタから抽出できる同出現頻度の特徴的フレーズの中から 4 種類を例に取り，それぞれを使用した際に取得できる SPARQL Endpoint 数を並べたものである．

表 6　Virtuoso クラスタの特徴的フレーズにおける取得 SPARQL Endpoint 数の違い

Virtuoso クラスタの特徴的フレーズ	取得 SPARQL Endpoint 数
“back to browser not saved”	59
“result can only be sent”	37
“allow you to retrieve remote”	43
“strict checking of void variables”	68

以上の通り，同じ単語数で同じ出現頻度のフレーズであっても，取得できる SPARQL Endpoint 数にはバラつきが生じる．また，取得 Endpoint 数のみを見た場合には最大値を示す”strict checking of void variables”を選出することが合理的であるように見えるが，実際には特徴的フレーズごとに取得できる SPARQL Endpoint が少しずつ異なっている．

そこで，3 つの類似 Web UI クラスタの特徴的フレーズから使用するフレーズをそれぞれ 4 種類まで増やし，それら特徴的フレーズをそれぞれ利用したフレーズ検索によって取得できる SPARQL Endpoint をマージすることによって，Linked Data リポジトリの収集性能がどのように変化するかを調べた．

表 7 は，3 種類の類似 Web UI クラスタについて，それぞれから抽出した 4 種類の類似フレーズを表している．また表 8 では，4 種類のフレーズ検索により取得した SPARQL Endpoint をマージした場合の取得 SPARQL Endpoint 数および，the Datahub 未登録の SPARQL Endpoint 数について表している．

表 7　クラスタごとの特徴的フレーズ

	特徴的フレーズ
Virtuoso クラスタ	“back to browser not saved”
	“result can only be sent”
	“allow you to retrieve remote”
	“strict checking of void variables”
SPARQLer クラスタ	“xslt style sheet blank for”
	“xml text csv xslt style”
	“set any options and press”
	“or use from in the”
RKB クラスタ	“above form is currently configured”
	“be made over the information”
	“information held within the repository”
	“repository using the sparql query”

表 8　複数の特徴的フレーズを用いた SPARQL Endpoint 取得

	取得 SPARQL Endpoint 数	the Datahub 未登録 SPARQL Endpoint 数
Virtuoso クラスタ	75	54
SPARQLer クラスタ	29	16
RKB クラスタ	42	8

表 5 と表 8 を比較すると，3 つのクラスタ全てにおいてより多くの SPARQL Endpoint を取得することができている．このことから，あるクラスタから複数の特徴的フレーズが抽出できる場合においては，複数の特徴的フレーズをそれぞれ個別に使用した SPARQL Endpoint 取得を行った上でその結果をマージすることで，より Linked Data リポジトリの収集能力を向上させることができると言える．

従って完全な自動収集システムとする上では，抽出できた特徴的フレーズを全て使用して SPARQL Endpoint を収集した場合に，最も良い性能を示すことができるものと考えられる．

また提案手法の収集能力について，精度の側面について触れる．ある特徴的フレーズを使用することで検索エンジンから取得できる検索結果には，SPARQL Endpoint ではない Web ページが含まれる可能性が常に存在する．しかし 4.4.2 節で述べたように，ある Web ページが SPARQL Endpint であるかどうかは，簡単な SPARQL 式を送信し応答を調べることによって判定することが可能である．従って，提案手法が収集した SPARQL Endpoint について，SPARQL Endpoint ではない Web ページが混入するということは考え難く，この点については 100%の精度を出すことができると言える．

6. おわりに

本研究では Linked Data リポジトリの自動収集を実現させることを目的として，既存のクローラ型検索エンジンを利用した自動収集手法を提案した．提案手法においては Linked Data リポジトリの Web UI が他のリポジトリの Web UI と Web ページとして類似するという性質に着目し，Structural Similarity および Content Similarity という 2 種類の類似性を用いた MDS による類似 Web UI 群の判別を行い，この類似 Web UI 群ごとに単語単位の 5-gram に基づいたフレーズの出現頻度測定を行うことによって，類似 Web UI 群ごとに共通して出現するような特徴的フレーズを抽出する．そしてこの特徴的フレーズを使用したフレーズ検索を既存のクローラ型検索エンジンにおいて行い，検索エンジンのクローラが収集した Linked Data リポジトリを取得するといった形で，本研究では Linked Data リポジトリの自動収集を実現させた．

この提案手法について，実際にサンプルとして the Datahub から取得した Linked Data リポジトリを使用し，自動収集の実験を行った．その結果，提案手法によって類似 Web UI 群ごとの特徴的なフレーズを抽出することができ，この特徴的フレーズを Google 検索のフレーズ検索に利用することで，the Datahub に登録のある既知のリポジトリおよび登録のない未知のリポジトリを収集できることを確認した．更に特徴的フレーズは類似 Web UI 群ごとに複数を抽出し併用することで，収集能力がより向上することを発見した．

加えて本研究においては，収集した Linked Data リポジトリに対する利用者の発見を支援するために，個々の Linked Data リポジトリを説明するような Web ページを関連ドキュメントとして収集し，この関連ドキュメントに対する全文検索とそれらに紐付けられた Linked Data リポジトリの提示による検索システムを提案した．この関連ドキュメントを利用した Linked Data リポジトリの検索システムが，ユーザの支援としてどの程度有効であるかについての検証が，本研究の課題として残されている．

謝辞

本研究を進めるにあたってご指導頂いた研究指導担当教員の阪口哲男准教授に感謝致します．また，主に合同ゼミの場で度々のご指導を頂きました副研究指導担当教員の永森光晴講師，ならびに杉本重雄教授，森嶋厚行教授に感謝致します．また，同じ研究室で協力して頂いた阪口研究室の皆様，合同ゼミでの発表において協力して頂いた杉本・永森研究室，森嶋研究室の皆様，同じ部屋で活発な議論にお付き合い頂いた鈴木伸崇研究室の皆様に，感謝致します．

参考文献

[1] W3C, "RDF - Semantic Web Standards," [Online].
Available: http://www.w3.org/RDF/.

[2] 神崎正英, セマンティック・ウェブのための RDF/OWL 入門, 東京: 森北出版, 2005, p. 11.

[3] "OpenLink Virtuoso Universal Server," [Online].
Available: http://virtuoso.openlinksw.com/.

[4] "AllegroGraph RDFStore　Web3.0's Database," [Online].
Available: http://franz.com/agraph/allegrograph/.

[5] "Welcome - the Datahub," [Online]. Available: http://datahub.io.

[6] "Data for Japan," [Online]. Available: http://dataforjapan.org/.

[7] "CKAN－The open source data portal software," [Online].
Available: http://ckan.org.

[8] "CKAN instances around the world | ckan - the open source data portal software," [Online]. Available: http://ckan.org/instances/#.

[9] "Sindice － The semantic web index," [Online]. Available: http://sindice.com.

[10] S. H. Heiko Paulheim, "Discoverability of SPARQL Endpoints in Linked Open Data," *CEUR Workshop Proceedings,* no. Vol 1035 (Proceedings of the ISWC 2013 Posters & Demonstrations Track), pp. 245-248, 2013.

[11] "Describing Linked Datasets with the VoID Vocabrary," [Online].
Available: http://www.w3.org/TR/void/.

[12] N. Steinmetz, H. Lausen and M. Brunner, "Web Service Search on Large Scale," *Service-Oriented Computing,* vol. Lecture Notes in Computer Science, no. 5900, pp. 437-444, 2009.

[13] R. Isele, J. Umbrich, C. Bizer and A. Harth, "LDSpider: An open-source crawling framework for the Web of Linked Data," in *9th International Semantic Web Conference(ISWC2010)*, 2010.

[14] T. Heath , C. Bizer, Linked Data Web をグローバルなデータ空間にする仕組み, 東京: 近代科学社, 2013, p. 7.

[15] M. Schmachterberg, C. Bizer and H. Paulheim, "State of the LOD Cloud 2014," 30 8 2014. [Online].
Available: http://linkeddatacatalog.dws.informatik.uni-mannheim.de/state/.

[16] 古谷野亘, 数学が苦手な人のための多変量解析ガイドー調査データのまとめかた, 東京: 川島書店, 1988, p. 150.

[17] "SPARQLer - An RDF Query Server," [Online]. Available: http://sparql.org.

[18] "RKBExplorer >>Projects >> ReSIST Resilence fro Survivability in IST," [Online]. Available: http://www.rkbexplorer.com/.

[19] Google, "重複するコンテンツ - ウェブマスター ツール ヘルプ," [Online]. Available: https://support.google.com/webmasters/answer/66359?hl=ja.

本研究の一部は，以下で発表している

- 瀬尾崇一郎，阪口哲男．“クローラ型検索エンジンを利用した SPARQL Endpoint 発見手法”．FIT2014(第 13 回情報科学技術フォーラム) D-01, Sep. 2014, http://www.ipsj.or.jp/event/fit/fit2014/index.html