スキーママッピングを基礎とした Web API と LOD の統合的利用環境の構築

A Development of a System to Combine Web API and LOD Dataset Based on Schema Mapping

阪本かすみ[1]　永森光晴[2]　三原鉄也[2]　杉本重雄[2]

Kasumi Sakamoto[1], Mitsuharu Nagamori[2], Tetsuya Mihara[2], and Shigeo Sugimoto[2]

[1] 筑波大学情報学群情報メディア創成学類

[1] Faculty of Library, Information and Media Science, University of Tsukuba

[2] 筑波大学図書館情報メディア系

[2] College of Media Arts, Science and Technologies, School of Informatics, University of Tsukuba

Abstract: On the web, numerous resources are not in the form of Linked Open Data (LOD). Lack of efficient means limits the seamless search of data across LOD and non-LOD resources. This study proposes a method for combining non-LOD resources with LOD by schema mapping in the Web API and the RDF data model. The use of the Web API facilitates to predefine the mapping from the published metadata schema.

1. はじめに

Web 上の情報が増加してきた現在、計算機が理解可能な形で情報を公開できる Linked Open Data(LOD)が注目されている。LOD として公開されているデータセットは SPARQL を用いて検索することができ、さらに横断検索機能である Federated Query[1]を用いることで複数の LOD データセットに問い合わせ、それぞれの結果を組み合わせた情報を得ることができる。LOD は様々な情報と組み合わせることでさらに利活用の場を広げている。

一方で、Web 上では LOD データセットとして存在しない情報が多く提供されている。例えば、検索サービスの Web API を活用することで天気などの動的データを取得できる。本研究では RDF 以外の形式で公開されているデータを非 LOD と呼ぶ。非 LOD と LOD の情報を組み合わせたアプリケーションを開発する時、それぞれの情報を別々に取得し、個別に組み合わせる必要がありコストがかかってしまう。SPARQL 式を通して Web API などが提供している非 LOD に問い合わせることができれば、LOD としては扱いづらい動的データとの連携も容易になり、LOD の利活用性の向上が期待できる。本研究では、Web API から取得した非 LOD と既存の LOD を繋げることを目的として、SPARQL 式から Web API へ問い合わせることができる環境の構築を行う。

2. LOD 利活用における非 LOD との結合の課題

SPARQL は LOD 利活用の場で欠かせないものである。LOD の普及に伴い SPARQL1.1 では様々な機能が追加された。その 1 つが Federated Query である。Federated Query では SERVICE 句を用いて、外部の SPARQL エンドポイントに対して問い合わせを実行でき、より多くの情報を取得できる。例えば、エンドポイント A に問い合わせる際、図 1 のように SPARQL 式を実行すると、エンドポイント A は SERVICE 句で指定されているエンドポイント B と C に問い合わせを実行する。利用者は 3 つのエンドポイントからの結果を組み合わせた情報を取得できる。Federated Query により、多様な情報を組み合わせることが容易になった。

LOD の増加と Federated Query を用いた横断検索により多くの情報にアクセス可能となっているが、Web 上に多くの非 LOD があるのも事実である。例えば、気象庁では過去の天気情報を CSV 形式で提供している[2]。また、警視庁は事件や事故の情報を公開し、犯罪情報マップとしてサービスを提供している[3]。これらの非 LOD には動的データなどが含まれており、比較的安定したデータが提供されている LOD データセットには存在しない情報が多くある。そこで本研究では、非 LOD と LOD を組み合わせる

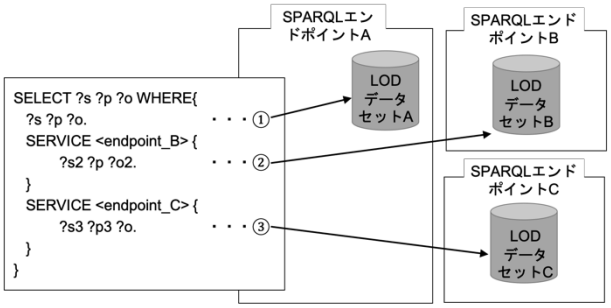

図 1　Federated Query の概要

ことで、今後の LOD の利活用性向上が期待できると考えた。しかし、機械的に LOD と非 LOD を結合するのは容易ではない。従来の非 LOD と LOD の結合方法は、それぞれのデータを別々に取得し、個別にデータマッピングを行うものであったため、コストがかかってしまっていた。これに対して本研究は、非 LOD の取得とマッピングのコストを削減するために、Web API ごとにあらかじめマッピングを定義しておく。これにより、SPARQL 式を通じて本システムに問い合わせることで、非 LOD と LOD の組み合わせを容易に行うことができる。

3. 関連研究

非 LOD を LOD 化する研究として RDF Mapping Language(RML)[4]がある。RML は様々な形式のデータを RDF 化するためのマッピング言語である。RML ではマッピングを定義するトリプルマップと呼ばれるものを生成し、トリプルマップもとに CSV、JSON などの様々なデータを RDF に変換できる。LOD の利活用のために非 LOD を用いている点では本研究と同じあるが、本研究とは異なりデータマッピングであり、利用者が非 LOD とマッピングを入力する必要がある。また、非 LOD の LOD 化であり本研究の目的とは少し異なる。

本研究と同じく SPARQL 式から非 LOD への問い合わせを可能にする研究として SPARQL Generate [5]が挙げられる。SPARQL Generate は SPARQL を拡張して、非 LOD データと RDF グラフのマッピングを SPARQL 式で記述することで、既存の LOD と合わせて非 LOD を扱うことができるシステムである。SPARQL Generate は本研究の目的と非常に類似している。しかし SPARQL Generate と異なり本研究では、対象となる非 LOD は Web API からの取得データである。また、スキーマベースでのマッピングを行うことで、利用者によるマッピングを削減する。

4. スキーママッピングを利用した LOD 化手法の提案

4.1 Web API と LOD の統合的利用

本研究では Web API を対象に、マッピングを用いて非 LOD の取得と RDF グラフの生成を自動的に行うシステムを構築する。利用者はマッピングを記述することなく、SPARQL 式を用いて Web API への問い合わせを実行でき、事前に用意されたマッピングに基づいて LOD 化された結果が取得できる。これにより、LOD と Web API にシームレスに問い合わせることができる。さらに双方の検索結果を組み合わせた情報を扱うことができる統合的利用を実現する。単純に非 LOD を LOD に変換するだけでなく既存の LOD と組み合わせることが、今後の LOD の利活用には重要だと考える。そこで本システムのアクセスに Federated Query を用いる。Federated Query を用いることで、利用者が SPARQL 式を実行する SPARQL エンドポイント上の LOD と本システムが変換した非 LOD を組み合わせた情報を提供することができる。

本システムの実現により、従来の方法で LOD データセットと Web API それぞれに対して問い合わせを実行し、取得したデータをマッピングして結合するまでの複数の過程を、SPARQL 式を記述する 1 つの作業で完結することができる。

4.2 スキーママッピングを利用した LOD 化手法

多くの Web API が問い合わせ方法や取得できるデータ構造などのメタデータスキーマを公開している。このメタデータスキーマを用いて Web API への問い合わせの設定および、取得データと RDF データモデルのマッピングを事前に用意することができ、自動的に行うことができる。このマッピングをスキーママッピング と呼ぶ。Web API から非 LOD を取得し LOD 化するまでの手順を以下に示した。

1) **Web API への入力値の取得**：Web API にアクセスし、非 LOD を取得するために、利用者の記述した SPARQL 式から問い合わせに必要な入力値を取得する。

2) **Web API への問い合わせ実行**：Web API ごとに決められた方法で URL を生成し検索を実行する。

3) **RDF グラフの生成**：取得した非 LOD から値を取り出し、それぞれの値を目的語として RDF グラフに格納する。

Web API の公開している情報からこれらの情報を取り出しスキーママッピングファイルとして用意しておくことで、それぞれの手順で必要な情報をそこから読み取り自動的に実行していくことができる。

4.3 Web API の分析

本研究を進めるにあたり、スキーママッピングの要求要件の調査を目的として、Web API の分析を行なった。分析には 2 万件以上の Web API に関する情報を検索・参照できるサービスである Programmable Web[6]を利用し、リクエスト方法とレスポンスデータ形式の観点でランダムに取り出した 25 件の Web API を分析した。

全ての Web API がリクエスト方法として GET メソッドを利用していた。その URL の構成としては、2 つのパターンがありそれぞれパラメータ形式とパス形式と呼ぶこととする。パラメータ形式は URL パラメータを用いたもので、キーワードなどの入力値からクエリストリングを生成し、ベースとなる URL に続けるものである。パス形式は、スラッシュ（/）で区切ったパスの設定により受け取る情報を決定するものである。また、この 2 つ URL パターンに加えて認証キーを採用しているものもあった。これは Web API が利用者に ID を与え、その ID がないと利用ができないものである。認証キーの対応に関しては、スキーママッピングの記述者が自らの ID をオープンにして利用可能にするか、利用者からの入力値とするかが考えられる。どちらの方法であってもスキーママッピングの記述者に一任することとする。

レスポンスデータ形式は、ほとんどの Web API が JSON 形式[7]か XML 形式[8]であった。JSON 形式のものが多かったため、本研究では JSON に対応するものを作成することにした。XML 形式など JSON 以外の形式でデータを提供しているものは一度 JSON 形式に変換したのち処理することで対応する。

4.4 SPARQL 式と Web API 問い合わせ URL のマッピング

Web API の分析から問い合わせ URL のパターンごとの構成要素は表 1 の通りである。この中で入力値以外の項目は、公開されている情報から分かるので事前に値を設定しておくことができる。固定値はできるだけベース URL に繋げた状態で設定しておくことで本システム内での処理を簡略化する。入力値は利用者から SPARQL 式を通じて取得する。

表 1　リクエストパターンにおける構成要素

リクエストパターン	ベースURL	パラメータ名	入力値	固定値
パラメータ方式	○	○	○	○
パス方式	○		○	○

Web API によっては複数の入力値があるのでそれぞれで以下の設定を行う。

- 値の説明
- パラメータ名
- 必須 / オプション
- 値の変換方法

利用者が正しい値を入力できるように、どのような値なのかという説明を提示する。利用者の記述した SPARQL 式から値を識別できるようにパラメータ名を設定する。全ての情報が URL の生成に必要なわけではないため、必須か否かの設定を設ける。クエリストリングに必要な入力値が特殊な形をとる場合があり、SPARQL 式から取得した入力値を加工しなければいけない。そのため値の変換方法に関して設定する。これらの情報をマッピングとして用意しておくことで、利用者が SPARQL 式で入力値を記述でき、それを用いて本システムは問い合わせ URL を生成できる。

4.5 レスポンスデータから RDF グラフを生成する

Web API から取得できるデータのほとんどが非 LOD であるので、マッピングにより RDF グラフを生成し LOD 化する。取得データと RDF データモデルのマッピングを行うためには以下の情報が必要である。

- レスポンスデータの形式
- 値を取り出すための JSON Path
- 値の変換方法
- 値を識別するためのパラメータ名

RDF グラフの目的語となる値を設定するために、取得データから値を取り出す。取得データが JSON データであると仮定すると JSON Path を用いて値を取り出すことができる。しかし、システム内で自動的に JSON Path を生成することはできないでの、それぞれの JSON Path はスキーママッピングに記述す

表 2　　SPARQL 式記述規則

	主語	述語	目的語
入力値	http://mdlab.slis.tsukuba.ac.jp/vocab #API名	http://mdlab.slis.tsukuba.ac.jp/input＃パラメータ名	対応する値
出力値		http://mdlab.slis.tsukuba.ac.jp/output＃パラメータ名	任意の変数
API一覧	http://mdlab.slis.tsukuba.ac.jp/vocab #service	http://mdlab.slis.tsukuba.ac.jp/mapping#APIlist	任意の変数
パラメータ一覧	http://mdlab.slis.tsukuba.ac.jp/vocab #API名	http://mdlab.slis.tsukuba.ac.jp/mapping#params	任意の変数

る必要がある。JSON 以外の形式は、変換ツールを用いて JSON に変換することで対応する。取り出した値のほとんどは数値かテキストであるが、LOD の利活用性向上を図るためにはこれらの値を既存の LOD データセットと繋げる必要がある。予め既存の LOD とのリンクを生成するメソッドを用意しておきスキーママッピングに値の変換方法として記述することで実現する。主語と述語は識別できるような値を自動的に生成し、取得したそれぞれの値でトリプルを形成していく。

5. Web API と LOD の統合的利用環境の構築

5.1 スキーママッピングを基礎とした統合的利用環境の概要

SPARQL 式から Web API へ問い合わせることができる環境を構築した。図 2 は本システムの概要図である。利用者は SPARQL 式内で SERVICE 句を用いて本システムへの問い合わせを記述する。SPARQL エンドポイントの SPARQL 式解析部分がそれを読み取り本システムに SPARQL 式を送る。本システムは受け取った SPARQL 式とスキーママッピングを用いて Web API へのクエリストリングを生成し実行する。その後、スキーママッピングを用いて返ってきた非 LOD を RDF グラフに格納し全ての情報をエンドポイントに返す。エンドポイントでは本システムの送った結果から利用者の求める情報を抜粋し、LOD の検索結果と組み合わせて利用者に提示する。

本システムに対して送る SPARQL 式に規則を与え、表 2 のように定義した。主語で使用する API を宣言し、入力値の設定は述語で問い合わせ時のパラメータ名、目的語でその値を記述する。出力値の設定は述語で取得したい値のパラメータ名、目的語で結果を格納する変数を記述する。これらの規則に加えて、利用者の使用を補助するために API 一覧とパラメータ一覧を表示する機能も作成した。本システムはこの規則に従った SPARQL 式から入力値を取得し、スキーママッピングを用いて Web API へ問い合わせる。

表 3　スキーママッピングの記述内容

INPUT	param		名前
	description		説明
	occurrence	mandatory	必須項目か否か
		optional	
	default		初期値
	value		値の加工方法
QUERY	base_url		ベースURL
	format		データ形式
	params	param	パラメータ形式のパラメータ名
		value	パラメータ形式の入力値
	path		パス形式の入力値を配列内に羅列
	other		URL末尾につく文字列
OUTPUT	param		名前
	path		値を取り出すためのJSON Path
	value		値の加工方法

5.2 スキーママッピングの記述方法

4 章で述べた要求要件に基づき、スキーママッピングの記述内容を表 3 のように定義した。マッピングは 3 つの項目から構成され、JSON 形式で記述する。

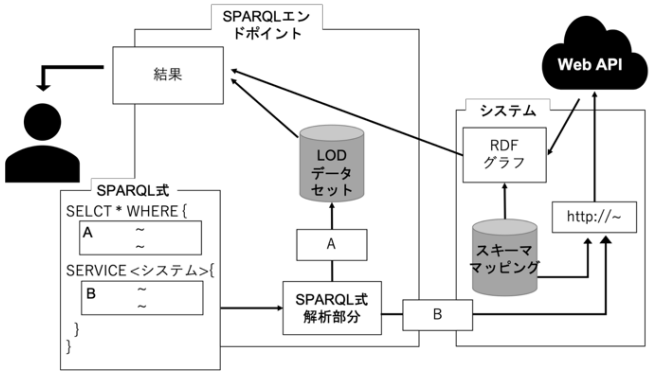

図 2 システム概要図

表 4　値変換メソッド

メソッド名	パラメータ	説明
get_uri_db(keyword)	キーワード(英語)	DBpediaとのリンク生成
get_uri_dbj(keyword)	キーワード	DBpedia Japaneseとのリンク生成
get_uri_yubin(postal)	郵便番号	郵便番号検索サイトのURLに変換
change_pluscodes([lat,long])	緯度,経度	Plus CodesのURLに変換
get_lat_tile_coordinate(lat)	緯度	地理院地図のタイルX座標に変換
get_long_tile_coordinate(long)	経度	地理院地図のタイルY座標に変換
get_areacode(area)	地域名	NHK番組表API独自のエリアコードに変換

INPUT には SPARQL 式で記述する入力値に関する情報を記述する。記述内容はそれぞれ param、description、occurrence、value と定義する。param は本システム内で値を処理するためのパラメータ名であり、記述者が任意の値を設定する。description は利用者が記述すべき入力値の説明を記述する。occurrence にはその入力値が必須か否かの設定で、必須の場合は“mandatory”、そうでない場合は”optional”と記述する。また、初期値が設定されている場合は default として記述する。value は Ruby スクリプトか用意されているメソッド(表 4)を用いて加工方法を記述する。加工しない場合は省略可能である。

QUERY には Web API の問い合わせ URL のマッピングに関して記述する。リクエストパターンによって記述内容が異なり、パラメータ方式の場合は、base_url、params、format の 3 つ、パス方式の場合は、base_url、path、other、format の 4 つを記述する。base_url には Web API 指定の URL に認証キーなどの固定値を加えたものを記述する。format には JSON、XML など結果のデータ形式を設定する。JSON 形式の場合は省略可能である。パラメータ形式の場合は params で base_url に続くパラメータ情報を記述する。1 つのパラメータに対して param で URL パラメータ名、value で値の指定を行う。params 内には、param と value を 1 つのオブジェクトとしてパラメータの数だけ記述する。パス形式の場合は path に base_url に続く順に値を設定する。URL 全体の最後が入力値で終わらない場合もあるので、その後に続く値を other に記述する。どちらの形式でも値の指定で、”@”に続けて INPUT で設定したパラメータ名を宣言することで、値を参照することができる。

OUTPUT には取得データと RDF データモデルのマッピングを記述する。記述内容をそれぞれ param、path、value と定義する。param は RDF データモデルの述語の設定に用いるパラメータ名としての役割と、INPUT 同様、本システム内で値を処理するためのパラメータ名として役割がある。path は Web API からの取得データから値を取り出すための JSON Path を記述する。value は取得データから取り出した値を self としてメソッドや Ruby スクリプトを用いて加工方法を記述する。加工しない場合は省略可能である。

利用者が記述した値を Web API が独自に設定している値に整形する必要がある場面がある。また、取得データのほとんどがリテラルであるが、統合的利用を実現するためにはできるだけ多くの LOD とのリンクが必要である。そのため INPUT と OUTPUT の値を変換できるメソッドを 7 つ用意した。表 4 にそのメソッド名と説明をまとめる。上の 3 つは INPUT で入力値の変換に用いるもので、下の 4 つは OUTPUT で出力値の変換に用いるものである。入力値の変換は現在対応している Web API で独自に設定されている値に変換するためのものである。出力値の変換はリテラルを URL に変換するものであり、既存の LOD に繋げるもの 2 つと Web サイトと繋げるもの 2 つを用意している。これら以外にも、経緯度から住所への変換や、住所から経緯度への変換などは使用頻度が高く、標準メソッドとして事前に用意しておく必要がある。新たに対応する Web API に合わせて随時増やしていく予定である。図 5 に Heart Rails Express[9]のスキーママッピングの例を示す。

5.3 Web API 問い合わせ URL の生成

本システムは記述規則に従って記述された SPARQL 式から使用する Web API とクエリストリングに使用する入力値、結果を格納するための変数を取得する。SPARQL 式が規則に従っていれば、図 3 のように使用する Web API 名、Web API への入力値、結果を格納する変数名が分かる。使用する Web API

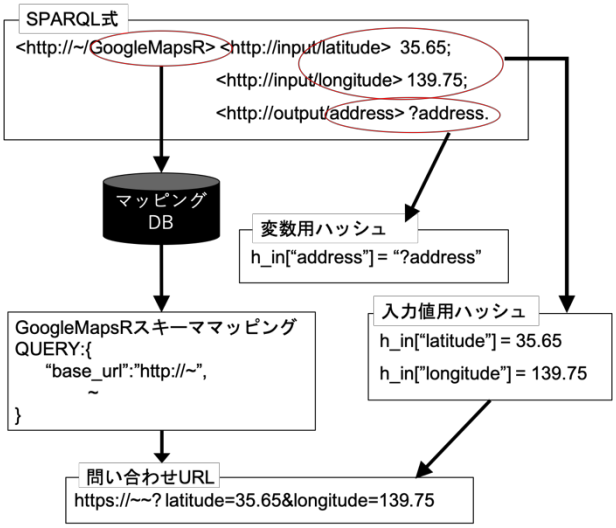

図 3　SPARQL 式から問い合わせ URL を生成

に対応するスキーママッピングを読み込み、入力値と変数はそれぞれハッシュに格納する。

次に、ハッシュに格納した入力値とスキーママッピングを利用して Web API クエリストリングを生成する。スキーママッピング INPUT を参考に、変換が必要な入力値は変換メソッド(表 4)を利用して変換する。スキーママッピング QUERY を参考に、リクエスト方式にあったクエリを生成する。パラメータ方式であれば、すべてのパラメータにおいて「パラメータ名=入力値」の形を作り、それぞれを”&(アンド)”で結びつける。パス方式であれば入力値、”/(スラッシュ)”の順につなげ、最後にそのほかの値(other)をつなげる。生成したクエリをベース URL につなげて Web API 問い合わせ URL が完成する。

5.4 取得データの RDF 化

完成した問い合わせ URL を実行し、Web API から非 LOD が取得できる。このデータを本システム内で処理をするために構文解析する。現在対応しているのは JSON 形式と XML 形式であり、XML 形式の場合は構文解析をした後、JSON 形式に変換する。スキーママッピング OUTPUT に記述されている JSON Path を用いて解析したデータから値を取り出す。変換が必要な値はスキーママッピングに記述されている通りに変換する。メソッド以外でも文字列の操作など簡単な処理は Ruby スクリプトで記述されていれば変換できる。次に値を目的語として RDF グラフを生成する。主語と述語は図 4 のようにそれぞれ、ID を付与したものと、パラメータ名を含むものを生成する。全ての値に関してこのトリプルを RDF グラフに格納する。最後に、作成した RDF グラフから全

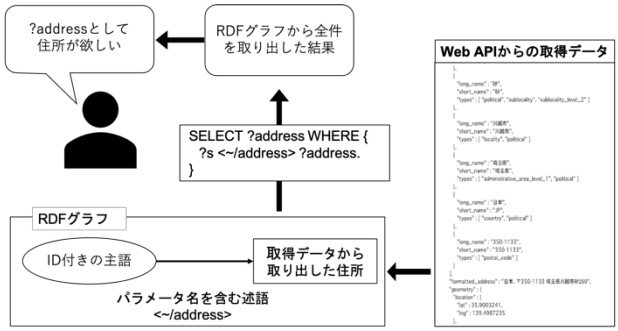

図 4　取得データの RDF 化

件を取り出して、利用者の指定した変数に入れて返す。Federated Query では、渡されたデータの中から利用者の指定した変数に格納されている値のみを選択して提示する。本システム内での処理を容易にするため、この機能を用いて求められていない情報も含めて全ての値を RDF グラフから取り出す。これにより、SPARQL 式で求められた非 LOD を LOD 化し、既存の LOD と組み合わせて利用者に提供することができる。

6. 評価実験

本研究で構築した本システムが、SPARQL 経験者にとって容易に利用できるシステムであることを評価するために実験を行った。SPARQL 経験者 7 名に対して以下の手順でアンケートと課題を実施した。

<実験手順>

1. 被験者の SPARQL 習熟度を調査するためアンケートを行う
2. 本システムの利用方法を説明する
3. 本システムを利用せず、Web 上にある情報から課題 3 つを行う　・・・①
4. 本システムを利用して課題 3 つを行う　・・・②

<課題>

1. 明後日の朝 6 時台に札幌の NHK 総合 1 (g1)で放送される番組を調べてください
2. 茨城県つくば市天王台 1-1-1 の明日 12:00(正午)の天気を調べてください
3. 東京都文京区後楽 1-7-27 の最寄り駅との距離と最寄り駅の前の駅を調べてください

アンケートでは、SPARQL の使用頻度と知識量から習熟度を測る。また、本システムの利用者は Web API そのものを利用する訳ではないが、問い合わせ

表 6　②の解答時間

被験者ID	課題1	課題2	課題3
001	7分20秒	6分5秒	7分46秒
002	4分15秒	6分24秒	8分30秒
003	7分57秒	7分49秒	3分53秒
004	8分	6分42秒	2分45秒
005	5分40秒	7分10秒	2分57秒
006	5分13秒	6分8秒	3分58秒
007	7分14秒	7分16秒	3分16秒

方法などの知識の有無によって SPARQL 式の記述に影響があると考え、Web API の利用の有無を合わせて調査する。アンケートから被験者全員が SPARQL に関して基礎的な知識があることがわかった。SPARQL 式の使用頻度、学習期間、知識量の点で分けると、初級 3 名、中級 2 名、上級 2 名であった。SERVICE 句の使い方を知らないと答えた被験者には､本システムの利用方法と合わせて SERVICE 句の使用方法も説明した。本システムの利用方法説明では、5.1 節の SPARQL 記述規則と機能の説明をし、それらを用いて一度チュートリアルを行い動作の確認をした。①の Web 上での検索は②の本システム利用の比較実験として行った。これにより、本システムの利用で SPARQL 式からの非 LOD への問い合わせがどの程度再現できているかを知ることができる。①では Web 上の検索であれば検索方法は自由とした。3 つの課題は本システムを利用することを前提として設定しており Web API を利用した方が正しい結果を導きやすい場合もあるため、①においても Web API の利用を許容した。①では検索方法によって②の検索結果と異なる結果になる場合があるが、検索方法に問題がなく結果が②のものとかけ離れていない限り検索完了とみなした。また、②ではあらかじめ PREFIX(接頭辞)を設定しておき、主語と述語を記述しやすい環境で行った。これは単にスムーズに実験を進めることができるための設定であり、使いやすさには影響しないと考える。

表 6 はそれぞれの被験者の②における解答時間である。アンケート結果と表 6 から Web API の利用経験の有無や SPARQL の使用頻度、SERVICE 句の理解度は本システムの利用に影響しないと考えられる。基本的な SPARQL 式の記述方法を知っていて、本システムの利用方法を理解していれば容易に利用することができる。また、ほとんどの被験者が利用を重ねるにつれて解答時間が短くなっている。このことから数回の使用で利用方法を十分に理解し、活用できると言える。

表 7　①と②の解答時間と問い合わせ回数の平均

平均	①解答時間	①問い合わせ回数	②解答時間	②問い合わせ回数
課題 1	1分16秒	4.4	6分31秒	4.7
課題 2	1分24秒	5.1	6分48秒	4.1
課題 3	3分55秒	7	4分43秒	3.3

次に、表 7 に課題の解答時間および問い合わせ回数の平均を示す。なお、①における問い合わせ回数はページの遷移回数であり、②における問い合わせ回数は実行ボタンのクリック数とする。①に比べて②は平均的に解答時間が長いが、問い合わせ回数は同じくらいであった。課題実施時には多くの被験者が説明文とパラメータ名一覧を参照しながら SPARQL 式を記述していた。このことから、SPARQL 記述規則に即した SPARQL 式の記述に時間がかかってしまったと考えられる。①と②の解答時間を比較すると、全ての課題において本システムを利用して検索をするよりも、Web 上で検索した方が早く結果までたどり着いている。しかし、①において課題 1 および課題 2 と比べて課題 3 は検索するのが困難であり時間がかかっていた。一方で、②ではどの課題においても一定の時間で結果にたどり着くことができていた。これは本システムを利用することで複雑な検索であっても単純な検索と同程度のコストで実行できると言える。

SERVICE 句と SPARQL 式の記述規則の説明だけで、全ての被験者が本システムを用いて結果にたどり着くことができ、問題なく利用できた。本システムのスムーズな使用には利用方法の十分な理解が必要であるが、実験では本システムの使用回数に伴い解答時間、問い合わせ回数ともに減少する傾向にあり、利用方法は理解が容易ですぐに実践できるものであると言える。以上から、本システムの利用により SPARQL 経験者であれば容易に Web API へ問い合わせることができると言える。

7. おわりに

多くの情報が LOD データセットとして公開されている一方で、LOD データセットとして公開されていない情報が Web 上に多く存在する。しかし、これらの非 LOD と LOD を組み合わせて利用するためにはコストが大きく、効率的に双方の情報を取得して結合できる環境は整っていない。これに対して本研究では Web API のメタデータスキーマを用いて RDF

データモデルとのマッピングを事前に記述することで非 LOD と LOD を統合する手法を提案した。これを実現するために、SPARQL 式から Web API へ問い合わせ、取得した非 LOD と既存の LOD を組み合わせることができる環境を構築した。現在本システムは 9 つの Web API に対応している。実験では多くの被験者が問題なく問い合わせを行うことができ、SPARQL 経験者であれば容易に本システムを利用できると言える。しかし実験後の聞き取りで、パラメータ情報の取得が大変であるという意見が多くあった。Web API の情報を知らない利用者にとって、SPARQL 式を記述するためにはパラメータ一覧は常時必要となる情報である。そのため、パラメータ一覧を表示しつつ SPARQL 式を記述できるアプリケーションを構築する必要がある。

今回の実験は SPARQL 経験者を対象として実験を行なったが、本システムの利用に必要な SPARQL の知識を測るために SPARQL 未経験者を対象として追加実験を行う予定である。今後は、対応する Web API を増やすことと、本システムの利用環境の検討が必要である。現在は GET メソッドにのみ対応しているので、POST メソッドへの対応も目指す。入力値として経緯度を用いる Web API は多くあるが、利用者の使い勝手を考慮すると住所を用いて問い合わせ可能であることが必要であるので、基本の変換メソッドとして逆ジオコーディングメソッドを用意する必要がある。このように、頻繁に必要とされ得る変換は基本メソッドとして用意する必要がある。

謝辞

本研究の一部は科研費 18K11984 の助成による。

参考文献

[1] “SPARQL 1.1 Federated Query”. World Wide Web Consortium. 2013-03-21.

[2] “過去の気象データ・ダウンロード”. 気象庁. https://www.data.jma.go.jp/gmd/risk/obsdl/index.php, (参照 2019-02-05).

[3] “事件事故発生マップ”. 警視庁. 2018-10-05. http://www.keishicho.metro.tokyo.jp/smph/jiken_jiko/hassei/map_annai.html, (参照 2019-02-05).

[4] RML Generic Mapping Language. http://rml.io/, (参照 2019-01-23).

[5] Maxime Lefrançois, Antoine Zimmermann, Noorani Bakerally. “A SPARQL extension for generating RDF from heterogeneous formats”. Extended Semantic Web Conference. Portoroz, Slovenia, 2017-05.

[6] Programmable Web. https://www.programmableweb.com/, (参照 2019-01-23).

[7] JSON の紹介. https://www.json.org/json-ja.html, (参照 2019-01-23).

[8] “Extensible Markup Language (XML) ”. World Wide Web Consortium. 2008-11-26. https://www.w3.org/TR/xml/, (参照 2019-01-23).

[9] “API”. Heart Rails Express. http://express.heartrails.com/api.html, (参照 2019-01-23).

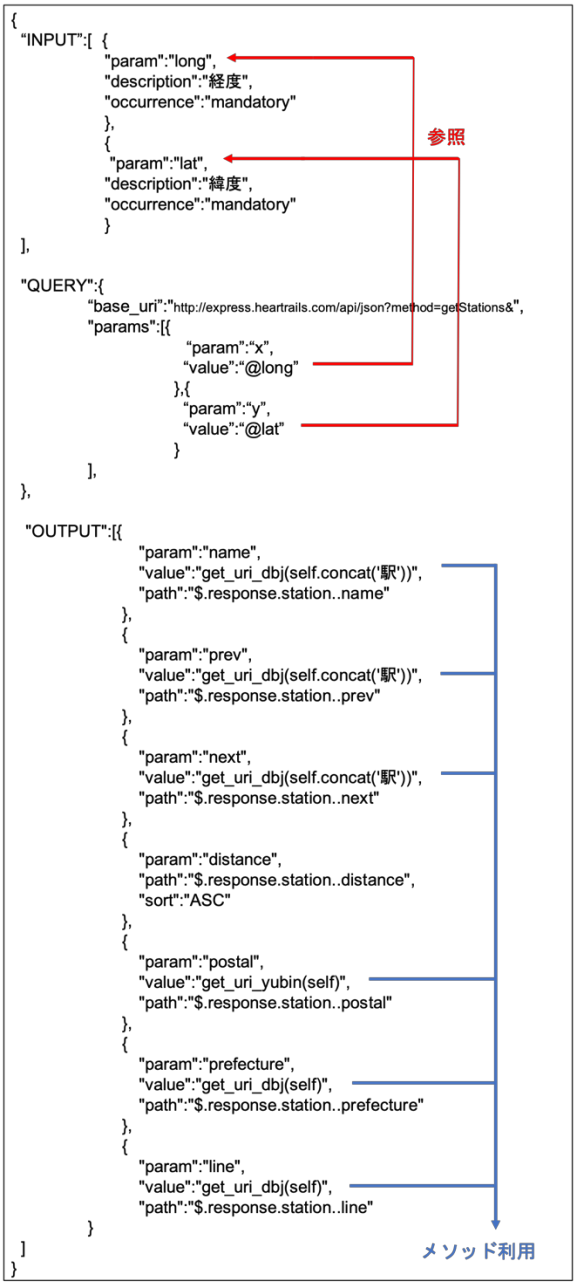

図 5 Heart Rails Express スキーママッピング