Linked Open Dataによる 店舗POIの公開と活用手法の提案

一円 真治[1] 梶 克彦[1] 廣井 慧[2] 河口 信夫[1,2]

概要：近年，飲食店・観光地といった POI(Point Of Interest) の情報を検索可能なサービスが多く提供されている．得られた POI の情報は，ユーザは今後の行動を決定する上で重要な要素であると考える．しかし，Foursquare のような POI 検索サービスには，各サービスが提供している方法でしか検索できなく，ユーザが欲しい情報を取得できないという問題点がある．本研究ではこの問題点に対し，ユーザの状況に応じた多様な検索クエリによる自由な検索を実現するため，複数の評価軸で店舗 POI 検索を可能にする店舗情報活用プラットフォームを提案する．データ構造に拡張性を持たせるため，データ共有技術である LOD(Linked Open Data) を本プラットフォームの DB に適応する．ユーザニーズにあった検索タグを選んでいくことで検索ストーリを自由に作り，それを元に LOD のクエリ言語である SPARQL を自動生成する．今回，SPARQL を自動生成する手法の評価のため，名古屋駅地下街の店舗に対しを用いて本プラットフォームの DB を構築し，検索内容に対して SPARQL を自動生成したところ，適切な情報を取得できることを確認した．

Proposal an Open and Utilization Method For Store POIs with Linked Open Data

Shinji Ichien[1] Katsuhiko Kaji[1] Kei Hiroi[2] Nobuo Kawaguchi[1,2]

1. はじめに

近年，スマートフォンの普及により，いつでもどこでもインターネットに接続し，観光スポット・飲食店といった POI(Point Of Interest) の情報を現地に行くことなく調べることが可能となっている．インターネットから得られた POI の情報は，ユーザに興味を持たせ，次の行動へのきっかけや決め手になり，ユーザの意思決定に重要な要素である．現在では，POI の情報提供を行うサービスが多く登場している．例えば，飲食店検索サービスの食べログ [1] や，ユーザの現在位置周辺にある POI を推薦する Foursquare[2] 等がある，また，デパートや駅といった大規模な商業施設は Web ページを作成することで施設全体の POI 検索サービスを提供していることも多い．これらの POI 検索サービスには，ユーザは各サービスが提供している方法でしか検索できず，ユーザは状況に適した情報を取得できないという問題点がある．

本研究では，ユーザの多様な検索クエリによる自由な検索を実現するため，複数の評価軸で店舗 POI 検索を可能とする店舗 POI の検索プラットフォームを提案する．本プラットフォームの POI DB(Database) のデータ構造には LOD(Linked Open Data) の表現形式である RDF(Resource Description Framework) を採用する．DB を LOD として活用する目的は 3.3.1 項で述べる．各ユーザの検索したい事柄を表す検索ストーリを LOD として表したユーザ LOD を元に，検索ストーリに対応したクエリ自動生成手法を提案する．検索ストーリとは，ユーザの現在の状況を踏まえた検索したい事柄を自然言語で表した文章と，検索内容の要素を示す検索タグのセットからなる物と定義する．検索タグについては 3.5 節で述べる．

本稿では，2 章で RDF からの情報取得に関する研究と POI の情報検索を行う関連サービスについて紹介し，3 章

[1] 名古屋大学大学院工学研究科
Graduate School of Engineering, Nagoya University
[2] 名古屋大学 未来社会創造機構
Institute of Innovation for Future Society, Nagoya University

では店舗 POI 情報活用プラットフォームを提案する．4 章では，提案プラットフォームの POI DB を名古屋駅地下街の POI データセットを用いて実装し，SPARQL 自動生成手法の評価結果について，5 章で本稿のまとめと今後の課題について述べる．

2. 関連事例

2.1 RDF からの情報取得に関する研究

RDF はデータの関係を主語，述語，目的語の 3 つの要素 (トリプル) で表現する．トリプルの集合は RDF グラフと呼ばれ，グラフ構造となっている．飯塚ら [3] は，全体の RDF グラフのデータからグラフ構造の類似度が高い特徴グラフを抽出し，抽出した特徴グラフを用いて SPARQL の自動生成方法を提案している．しかし，SPARQL を自動的に生成できたとしても，ユーザからの多様なクエリには対応できていない．武田ら [4] は，RDF のデータ構造を理解のために，RDF に対し，情報を辿るようにして情報検索を行う探索的検索を可能にするアプリケーションを開発している．このアプリケーションにより，探索的検索を行うには，RDF はトリプルで構成されていること，用いられている語彙等 RDF や LOD に関する知識を必要とするという問題がある．

2.2 POI の情報検索を行うサービス

店舗検索サービスの中でも，飲食店検索に特化したサービスとして Hot Pepper[5]，食べログ等がある．また店舗に限らない POI 全体の検索サービスは，Google Maps[6] や Yahoo!地図 [7] といった地図検索サービスの機能の一つとして提供されている．これらの検索サービスで POI 検索を行う場合，ユーザの調べたい事例に対応したキーワードの選択や，入力が必要となる．また，選択できる項目やその組み合わせは，サービスごとに限られているため，ユーザの検索はある一定の範囲に制限されているという問題がある．食べログを例に，この問題点について説明する．食べログは，ユーザの口コミ情報をベースにしており，ジャンル・住所等で情報を絞り込み，料理のおいしさスコアや雰囲気スコアといったユーザスコアを評価軸とし順序付けを行うことで，飲食店をランキング形式で検索できるサービスである．食べログは他のサービスと比較して多くの絞り込み項目を提供しているが，ユーザの求めている項目がなければ，適切な検索方法が分からないため，知りたい情報にたどり着くのは困難である．また，ユーザの位置・時間といったコンテキスト情報や性別・年齢といったデモグラフィック情報といったユーザメタ情報の組み合わせによる検索シーンのパターンは膨大にあるため，ユーザスコア以外の方法でも情報の評価が可能であるべきであると考える．例えば，ユーザの現在位置から近い順であったり，扱っているメニューの種類の多さ，カロリー量，年齢層でもランキング等である加えて，複数の評価軸を組み合わせて情報の順付を行うことが可能であるべきである．

3. 店舗情報活用プラットフォームの提案

提案するプラットフォームの全体図を図 1 に示す．本プラットフォームは 3 つのレイヤで構成されている．第一に，利用者であるユーザレイヤ，第二に，ユーザレイヤからの情報を処理するシステムレイヤ，第三に，POI DB や他の LOD とのつながりにより大規模な DB を構築する LOD クラウドレイヤがある．本章では，まず本プラットフォームの必要性についてと動作の流れ・開発ステップについて述べ，その後各レイヤについての詳細を紹介していく．

3.1 提案プラットフォームの必要要件

店舗や観光スポット等の既存の検索サービスでは，エリアやキーワード，ジャンルといったサービス側から提供されている項目を選び，要素を選択・入力することで，検索クエリを作成している．つまり，ユーザは，限定された条件でしか検索を行えない．しかし，POI に関連付く情報は多様に存在すると考える．POI に関連付く情報の例としては，位置情報や住所情報，ジャンルといった客観的に一意に決まる情報に加えて，ユーザ評価スコアやセール情報・チェックイン数・口コミ情報等のユーザが生成する情報，POI の管理者側が提供する情報がある．多様な POI の情報に対応するため，拡張性のあるデータ構造をもち，ユーザがより自由な検索を可能とする POI 情報活用プラットフォームが必要である．POI に対して，情報の拡張を容易にするために POI DB に LOD の技術を適用させる．提案するプラットフォームは，LOD 化した POI を情報拡張・検索・推薦の 3 つの面で効率的に活用するためのプラットフォームである．

3.2 動作の流れ・開発ステップ

3.2.1 動作の流れ

ユーザは，検索したい事柄を表現する検索ストーリをシステムレイヤの検索ストーリ入力インターフェースにより作成または選択し，システムレイヤのクエリシステムにリクエストを送る．クエリシステムはリクエストである検索ストーリを構成する検索タグセットから，LOD クラウドレイヤの POI DB に送る SPARQL を自動的に生成し，該当するレスポンス結果を得る．そして，その結果をシステムレイヤのアプリケーションが受け取り，ユーザが求める形式で提供・可視化を行うという流れになっている．

ユーザが検索ストーリを選択する際の，3 つのケースを以下に示す

(1) これまでにユーザ LOD に登録されている検索ストーリを再度選択するケース
(2) ユーザ LOD は各ユーザが持っているデータであるの

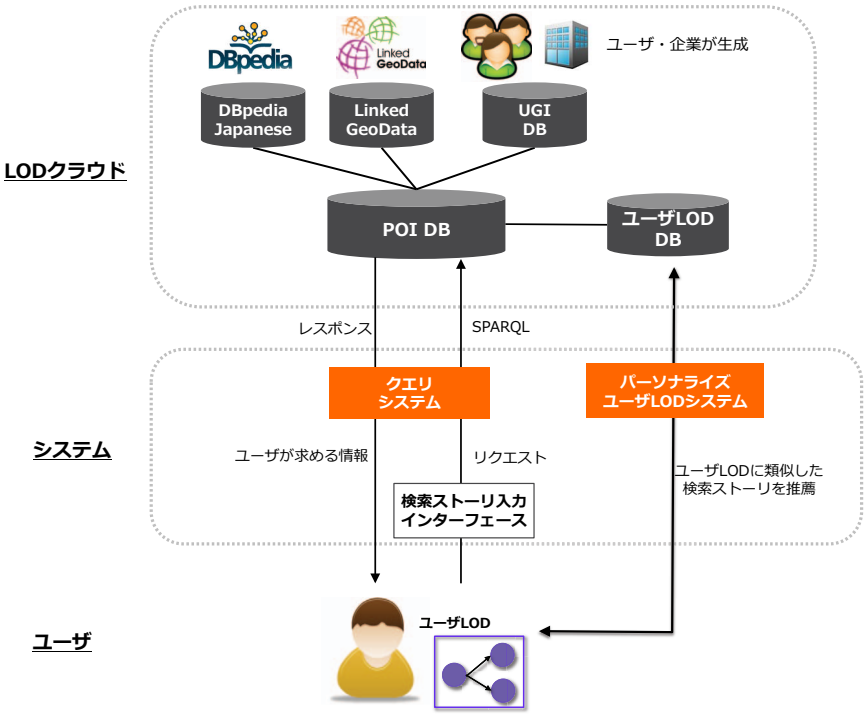

図 1　店舗 POI 情報活用プラットフォーム

で，他のユーザのユーザ LOD との類似性から，似ているユーザを推定し，そのユーザの検索ストーリから選択するケース

(3) 検索頻度の高い検索ストーリをランキングから選択するケース

(2) は，趣味・年齢・性別といったユーザメタ情報が似ているユーザは，同じような検索をする可能性があるという協調フィルタリングの考えに基づいている [12]．新しい検索ストーリの推薦によって，ユーザに新たな情報の発見・気付きを与え，セレンディピティをもたらす可能性が高くなる．

3.2.2　開発ステップ

本プラットフォームの実用化に向けての開発ステップを以下に示す．

ステップ 1：　LOD クラウドレイヤの POI DB とシステムレイヤのクエリシステムの実装

ステップ 2：　他の LOD とのつながりの構築による DB の拡張

ステップ 3：　ユーザレイヤの複数の評価軸によるレスポンス情報可視化アプリケーション

ステップ 4：　ユーザ LOD を用いた検索ストーリ推薦システム

3.3　LOD クラウドレイヤ

LOD クラウドレイヤには，本プラットフォームのメイン DB であり，かつ他の LOD とのハブの役割を果たす POI DB がある．POI DB とは，POI の基本的な情報とそれに紐づく動的な情報を管理する DB である．POI の基本情報は，名前情報・位置情報・ジャンル (例：飲食店，アパレルショップ，本屋) といった客観的に一意に決定できる情報 (スタティックな情報) であり，動的な情報とは，営業時間・メニュー・価格等，時間によって変化する可能性のある情報であると定義する．POI DB ではそれらの情報を LOD として利用するために RDF で保存する．データ構造については 3.4 で詳しく述べる．

3.3.1　LOD を活用する意味

LOD を活用するメリットを以下に示す．

- データスキーマは，RDB(Relational Database) のようなテーブル構造でなくグラフ構造であるため柔軟なデータスキーマを構築可能．
- 他の LOD とのつながりにより，DB を拡張でき，大規模な DB が構築可能．

LOD は RDF のグラフ構造により，表現され，情報のリソースであるノードとノード間の関係性を示す有向リンクで構成される．あるノードに情報を追加したい場合，関連付けるノードとそのリンクを追加する．また，他の LOD とのリンクを容易に構築可能な点に，LOD の最大のメリットがあると考える．LOD として Web 上に情報を公開することで，新しく DB を作成する必要がなくなり，容易に DB を拡張できる．このように，データのつながりにより，DB の知識を拡張可能となる．既存の LOD の例としては，日本語 Wikipedia を LOD 化した DBpedia Japanese[9]，地

表 1　語彙まとめ

Prefix	URL	作成
rdfs	http://www.w3.org/2000/01/rdf-schema#	
rdf	http://www.w3.org/1999/02/22-rdf-syntax-ns#	
lisra	http://lisra.jp/resource	○
lisras	http://lisra.jp/schema#	○
lisramenu	http:///lisra.jp/menu#	○
xmls	http://www.w3.org/2001/XMLSchema#	
gr	http://purl.org/goodrelations/v1#	

理情報の LOD である Linked Geo Data[11]，学術情報を LOD 化する LODAC[10] 等がある．本研究では，提案プラットフォームを実現し，Open Data の価値を示し，更に Open Data の活発化も目的としている [8].

3.4　提案する POI のデータ構造

提案する店舗 POI のメタデータ構造の図 2 に示す．各 POI は ID で管理されており，POI に関連する情報はノード (オブジェクトリソース) とその間の関係を示すプロパティ (リンク) とで表現する．POI ID は，lisra-schema という独自で定義した語彙で定義してあるクラスのサブクラスであり，検索の際に，検索タグとなり得るプロパティは，lisras:FilterProperty，または lisras:RankProperty のサブクラスである．また，これらのサブクラスでないプロパティとノードを接続することも可能である．本プラットフォームで使用する語彙については，表 1 にまとめた．作成の項目に該当する語彙は，我々が独自に定義した語彙である．

店舗 POI のデータ構造の具体例として飲食店 POI のデータ構造を図 3 に示す．POI ID に間接的にでも関連付くインスタンスは，すべて POI ID に関連付けることとする．階層構造を表現する場合について，図 3 を例に説明する．図 3 では，営業時間インスタンスである営業時間 ID からメニューリストのインスタンスであるメニューリスト ID にリンクを張る．このようにインスタンスからインスタンスにリンクを構築する．このようにするのは，検索クエリの生成を簡易化するためである．検索クエリの生成については 3.6 節に述べる．

3.5　ユーザレイヤ

ユーザは検索したい事柄を表現する検索ストーリをシステムレイヤの検索ストーリ入力インターフェースを用いて作成する．検索ストーリに対する検索タグの組み合わせの例を表 2 に示す．検索タグとは，ユーザの検索したい条件を示すタグのことである．検索タグには，情報の絞り込みのためのフィルタタグと情報の順序付けのためのランクタグの 2 種類が存在する．フィルタタグは，lisras:FilterProperty のインスタンスであるプロパティとそれに関連付くリソース値の組を指し，ランクタグは，lisras:RankProperty の

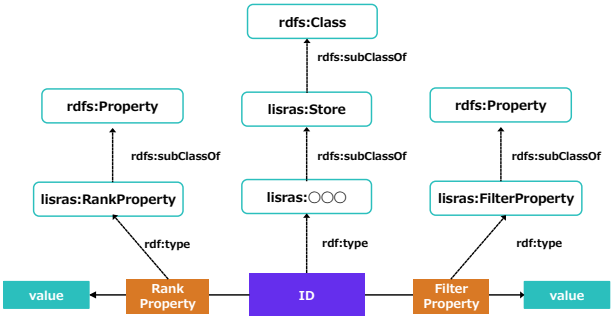

図 2　店舗 POI のデータ構造の図

表 2　検索ストーリに対するタグの組み合わせ

検索ストーリ	検索タグ
名古屋駅内のラーメン屋 を現在位置から近い順に知りたい	フィルタ：名古屋駅内 フィルタ：ラーメン フィルタ：飲食店 ランク：位置情報

```
prefix rdfs:  <http://www.w3.org/2000/01/rdf-schema#>
prefix lisra: <http://lisra.jp/resource/>
prefix lisras: <http://lisra.jp/schema#>
prefix xmls:  <http://www.w3.org/2001/XMLSchema#>
prefix rdf:   <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
prefix gr:    <http://purl.org/goodrelations/v1#>
prefix lisramenu: <http://lisra.jp/menu#>

select distinct ?filterProperty ?filterPropertyResource   where{
?filterProperty a lisras:FilterProperty.
?s ?filterProperty ?filterPropertyResource.
}
```

図 4　フィルタタグを取得する SPARQL

```
prefix rdfs:  <http://www.w3.org/2000/01/rdf-schema#>
prefix lisra: <http://lisra.jp/resource/>
prefix lisras: <http://lisra.jp/schema#>
prefix xmls:  <http://www.w3.org/2001/XMLSchema#>
prefix rdf:   <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
prefix gr:    <http://purl.org/goodrelations/v1#>
prefix lisramenu: <http://lisra.jp/menu#>

select distinct ?rankProperty where{
?rankProperty a lisras:RankProperty.
}
```

図 5　ランクタグを取得する SPARQL

インスタンスであるプロパティを指す．また，各タグはユーザの状況を表す「コンテキスト」，検索ワードとしての「キーワード」，順序付けのための「評価」の 3 種類の内のいずれかの性質を持っているものとする．

3.5.1　ユーザ LOD

ユーザ LOD を図 6 に示す．ユーザ LOD は，UID(UserID) を中心にユーザメタ情報と検索ストーリで構成されている．ユーザメタ情報は，UID に直接リンクが張ってあり，検索ストーリは，ストーリ ID に関連する検索タグにより構成されている．データ形式は RDF であるため，情報の追加があれば，リソースとプロパティを追

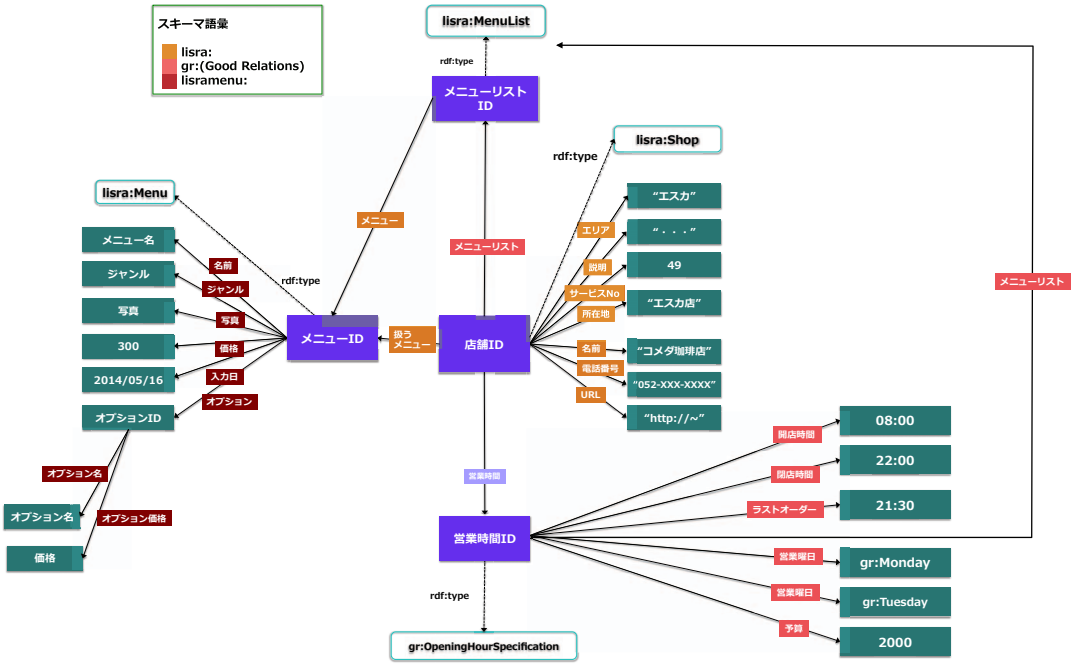

図 3　飲食店 POI のメタデータ構造の図

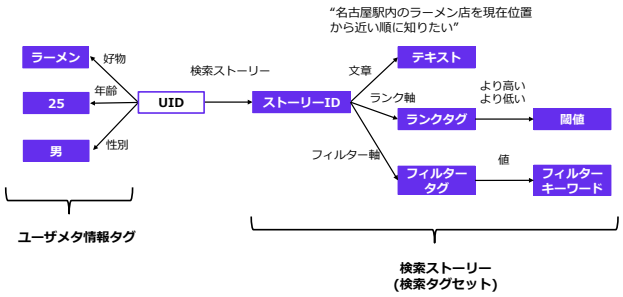

図 6　ユーザ LOD

加するだけで，ユーザ LOD を拡張することが容易に可能である．

3.5.2　ユーザ生成情報

本プラットフォームの DB は単一の DB ではなく，LOD クラウドを想定している．単一の DB であると，ユーザが活用できる情報に制限が生じる．LOD クラウドとすることで，複数の DB がつながり，情報の拡張ができる．

ユーザ生成情報とは，ユーザ自らが生成した情報を示す．例えば，店舗のレビューであったり，味・雰囲気・お店の混雑度といったスコア情報である．

また情報を集めるという観点では，クラウドソーシングの考えを取り入れているため，多くのユーザが本プラットフォームを活用し，少しずつ情報を増やしていけば，個人では現実的に収集不可能な情報でも DB の作成が可能である [14]．

3.6　システムレイヤ

システムレイヤには，検索ストーリを作成するインターフェースと作成された LOD のリクエストクエリである SPARQL を生成するクエリシステム，他ユーザのユーザ LOD との類似性から検索クエリの推薦を行うパーソナライズユーザ LOD システムが存在する．本節では，各システムについて述べる．

3.6.1　検索ストーリ作成インターフェース

ユーザは検索ストーリインターフェースとのインタラクティブな会話形式により，検索タグを選択，ランクタグに対する閾値を入力する [15]．検索ストーリ入力インターフェースのイメージを図 7 に示す．ユーザは提示された情報 (何がしたい？) に応じて，該当する検索タグの選択を複数回行い，検索ストーリを作成する．各種類の検索タグ取得の方法について述べる．検索タグは，LOD クラウドから取得する．取得元の LOD DB を変更することによって，多種多様な検索タグを取得できる．フィルタタグは，lisras:FilterProprty をキーにそれのプロパティインスタンス群を取得し，それらに関連するリソース値を取得する．ランクタグは，lisras:RankProperty をキーにそれのプロパティインスタンス群を取得する．LOD クラウドにリクエストする SPARQL を図 4，5 に示す．

3.6.2　クエリシステム

LOD クラウドにリクエストする SPARQL を生成する流れを図 8 に示す．検索ストーリをクエリ生成部が受け取り，SPARQL を自動的に生成し，LOD クラウドにリクエストを送る．そのレスポンス結果をデータ整形部がユーザ

図 7　検索ストーリ入力インターフェースのイメージ

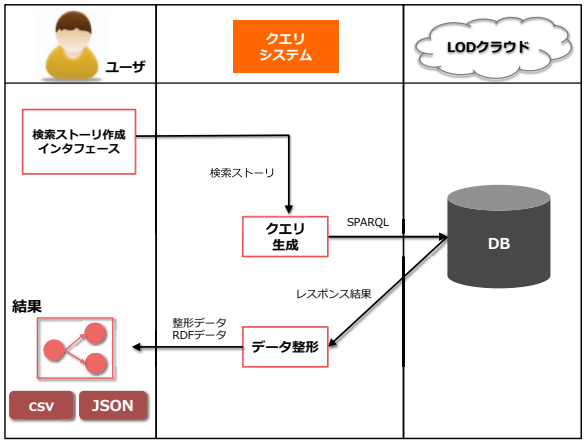

図 8　クエリシステムのシステムフロー

が求める形式 (例：JSON，XML，CSV) や，可視化クライアントアプリケーションが活用しやすい形式に変換する．

これより，各部について説明する．

クエリ生成：　検索ストーリを構成するフィルタタグとランクタグから，SPARQL を自動的に生成する．本プラットフォーム用の SPARQL のテンプレートを図 9 に示す．プロパティとリソース値から成るフィルタタグは，図 9 の 3〜4 行目の赤い部分に当てはめる．プロパティであるランクタグは，まず 4 行目の青い部分に当てはめ，各リソース (?FilterInstanceX) に関連付く値 (?value) を取得する．5 行目では，閾値を設定すれば，ランク値に対して閾値フィルタをかけることが可能である．6〜7 行目では，フィルタタグによって絞り込んだクラスインスタンスを用いて，目的情報 (?goal) を絞り込む．目的情報に順序付ける場合は，9 行目で昇順・降順にランクタグオプションを元に並び替える．フィルタ部とランク部は，各種のタグの個数

```
1  select distinct ?goal
2      where{
3          ?goal    フィルタータグによるフィルター .
4      ?FilterInstanceX    フィルタータグによるフィルター .
5      ?FilterInstanceY    ランクタグ    ?valueY.
6                  ?valueに対し，閾値フィルター
7          ?goal    ?pX    ?FilterInstanceX;
8                   ?pY    ?FilterInstanceY;
9      }          ランクタグオプション(昇順・降順)
```

図 9　本プラットフォーム用 SPARQL テンプレート

```
select distinct * where{
        ?goal  a  lisras:Restaurant .
        ?goal  lisra:area  "エスカ" .
        ?filterInstance1  lisramenu:price  ?value1.
          filter(?value1 < 800).
        ?goal  lisra:treatMenu  ?filterInstance1.
} order by  ?value1
```

図 10　自動的に生成された SPARQL の例

(数値：X，Y) に応じて変化し，目的情報絞り込みの部分もフィルタ部とランク部の合計インスタンス数 (X+Y) によって変化する．ランクタグオプション・閾値は，検索ストーリ作成の際に決定するものとする．検索ストーリから生成された SPARQL の具体例を図 10 に示す．赤い部分のフィルタタグから生成された部分は，lisras:Restraurant クラスのインスタンスでありかつ，エスカ (名古屋駅地下街エリアの一つ) に存在する POI に絞っている．青い部分のランクタグから生成された部分は，lisramenu:price プロパティを用いて，メニューの金額を?value に取得し，800 円を閾値にそれよりも安い金額の?filterInstance1 を抽出する．order by ?value を指定することで，金額の安い順に情報を返す．

提案する SPARQL テンプレートは，図 9 だけでなく，複数パターンを用意する．

データ整形：　LOD クラウドから得られたレスポンス情報を，可視化や閲覧用クライアントアプリケーションが処理しやすい形式に変換する．また，本プラットフォームから得た情報を利用して開発者がサービス開発を行うことも可能とするため，二次利用しやすい形式にも変換可能とする．

3.6.3　パーソナライズユーザ LOD システム

パーソナライズユーザ LOD システムのシステムフローを図 11 に示す．ユーザの検索ストーリと LOD クラウドのユーザ LOD DB と検索タグのマッチングを行い，類似度

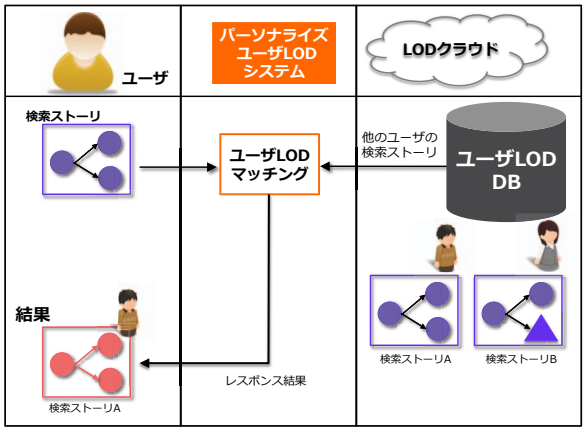

図 11　パーソナライズユーザ LOD システムのシステムフロー

表 3　名古屋駅地下街 POI データセットの概要

収集 POI 数	367
メニューリストがある レストラン POI 数	44
総トリプル数	9369

表 4　評価用検索ストーリ

パターン	検索ストーリ
A	地下街エリア（エスカ）にある飲食店で 20 時でも営業しており， 閉店時間が遅い順に知りたい
B	名古屋駅地下街全体で， 600 円以上・1000 円以下の麺メニューを 扱っている飲食店を知りたい

の高い他ユーザの検索ストーリをユーザに推薦する．

4. 評価・考察

4.1 評価方法

評価を行うため．名古屋駅地下街店舗 POI データセットを jena[17] を用いて RDF に変換し，POI DB には RDF ストアの Virtuoso[18] を使用し，POI DB を作成した．名古屋駅地下街店舗 POI データセットは，名古屋駅の地下街に関する Web ページ [16] から情報を収集し，作成した．飲食店に関しては，メニューデータが Web で公開されていた場合，メニューデータ収集している．収集した情報は，図 3 のデータ構造に対応している情報である．データセットの概要を表 3 に示す．評価としては，2 パターン行う．評価方法 1 は，作成した POI DB・Hot Pepper・食ベログに評価用検索ストーリに対応したクエリを各場合ごとに作成し，得られるレスポンス結果を比較する．方法表 2 は，POI DB に情報を追加した場合でも，テンプレートに当てはめて SPARQL を自動的に生成できるか検証する．

評価方法 1　評価方法 1 用の検索ストーリを表 4 に，各パターンに対する各ケースのクエリを表 5，表 6 にまとめる．検索タグから，自動的に生成された SPARQL を図 12，13 に示す．

評価方法 2　評価用に，POI DB に追加した RDF データ

表 5　各ケースのクエリ (パターン a)

ケース	検索タグ
検索システム	フィルタタグ：エスカ フィルタタグ：飲食店 ランクタグ：閉店時間 ランクタグ：遅い順
食べログ	エリア・駅：名古屋駅 (300m 周辺) キーワード：エスカ 営業時間：夜 10 時以降入店 OK
Hot Pepper	エリアキーワード：名古屋駅 店名：エスカ

```
prefix rdfs:  <http://www.w3.org/2000/01/rdf-schema#>
prefix lisra: <http://lisra.jp/resource/>
prefix lisras: <http://lisra.jp/schema#>
prefix xmls:  <http://www.w3.org/2001/XMLSchema#>
prefix rdf:   <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
prefix gr:    <http://purl.org/goodrelations/v1#>
prefix lisramenu: <http://lisra.jp/menu#>

select distinct ?goal  where{
 ?goal lisra:area "エスカ".
 ?goal a lisras:Restaurant.

 ?filterInstance1 gr:closes ?value1.
 ?goal ?p1 ?filterInstance1.

} order by desc( ?value1 )
```

図 12　パターン A 用の SPARQL

表 6　各ケースのクエリ (パターン B)

ケース	検索タグ
検索システム	フィルタタグ：麺類 フィルタタグ：飲食店 ランクタグ：600 円以上 ランクタグ：1000 円以下
食べログ	エリア・駅：名古屋駅 (300m 周辺) ジャンル：そば・うどん・麺類 予算：夜 1000 円
Hot Pepper	エリア：名古屋駅 料理：ラーメン 予算： 2000 円

```
prefix rdfs:  <http://www.w3.org/2000/01/rdf-schema#>
prefix lisra: <http://lisra.jp/resource/>
prefix lisras: <http://lisra.jp/schema#>
prefix xmls:  <http://www.w3.org/2001/XMLSchema#>
prefix rdf:   <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
prefix gr:    <http://purl.org/goodrelations/v1#>
prefix lisramenu: <http://lisra.jp/menu#>

select distinct ?goal where{
 ?goal a lisras:Restaurant.

 ?filterInstance1 lisramenu:genre "麺類".
 ?filterInstance2 lisramenu:price ?value2.
  filter(?value2 >= 600).
  filter(?value2 <= 1000).

  ?goal ?p1 ?filterInstance1.
  ?goal ?p2 ?filterInstance2.
}
```

図 13　パターン B 用の SPARQL

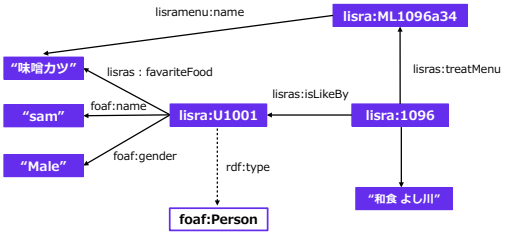

図 14　評価方法 2 用に追加した RDF データ

```
prefix rdfs:  <http://www.w3.org/2000/01/rdf-schema#>
prefix lisra: <http://lisra.jp/resource/>
prefix lisras: <http://lisra.jp/schema#>
prefix xmls:  <http://www.w3.org/2001/XMLSchema#>
prefix rdf:   <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
prefix gr:    <http://purl.org/goodrelations/v1#>
prefix lisramenu: <http://lisra.jp/menu#>
prefix foaf: <http://xmlns.com/foaf/0.1/>

select distinct ?goal   where{
 ?goal a lisras:Restaurant.

?filterInstance1 lisras:favariteFood "味噌カツ".

 ?goal ?p1 ?filterInstance1.
  ?goal lisra:name ?name.
}
```

図 15　評価方法 2 用に自動生成された SPARQL

表 7　検索結果 (パターン A)

	食べログ	Hot Pepper
適合店舗数	0	6
不適合店舗数	9	1
情報なし	0	22
合計	9	29

を図 14 に示す．lisra:1096，lisra:ML1096a34，”和食よし川”の 3 つのノードは予め POI DB に保存されている．用いた検索ストーリは「味噌カツが好きな人がお気に入りにしている店」である．検索タグは，「味噌カツ」,「飲食店」「お気に入り」である．自動生成した SPARQL を図 15 に示す．

4.2　結果

評価方法 1 における検索パターン A，B において，生成された SPARQL から得られた結果をそれぞれ図 16，17 に示す．食べログと Hot Pepper からの結果に関しては，パターン A は表 7 にパターン B は，表 8 に示す．得られた情報がどの POI であるのかを理解しやすくするため，図 16 では，閉店時間 (value1)，POI 名前 (name) を図 17 では，POI 名前 (name), メニュー ID(?filterInstance2)，メニュー ID に関連づく価格情報 (value2) を追加表示してある．自動的に生成した SPARQL で，提案するデータ構造を持つ POI DB から適切な情報取得が可能であると確認できた．

評価方法 2 の結果は，図 18 に示す．求める結果を得られたので，DB を容易に拡張可能であると確認できた．

goal	value1	name
http://lisra.jp/resource/1118	22:30:00+09:00	"マクドナルドハンバーガー名古屋エスカ店"
http://lisra.jp/resource/1175	22:30:00+09:00	"濃厚担担麺専門店想吃坦坦面エスカ店"
http://lisra.jp/resource/1093	22:00:00+09:00	"元祖手羽先唐揚風来坊"
http://lisra.jp/resource/1094	22:00:00+09:00	"まぐろ料理鮪小屋本店エスカ店"
http://lisra.jp/resource/1096	22:00:00+09:00	"和食よし川エスカ店"
http://lisra.jp/resource/1108	22:00:00+09:00	"ラーメン寿がきや名古屋エスカ店"
http://lisra.jp/resource/1110	22:00:00+09:00	"串揚げ珍串"
http://lisra.jp/resource/1113	22:00:00+09:00	"味噌煮込うどん山本屋本店エスカ店"
http://lisra.jp/resource/1114	22:00:00+09:00	"ステーキと焙煎カレーふらんす亭名古屋エスカ店"
http://lisra.jp/resource/1115	22:00:00+09:00	"新名古屋名物海老どて食堂"

図 16　パターン A 用 SPARQL の結果 (上位 1 0 件)

goal	name	filterInstance2	value2
http://lisra.jp/resource/1108	"ラーメン寿がきや名古屋エスカ店"	http://lisra.jp/menu#ML1108a21	"620"^^<http://www.w3.org/2001/XMLSchema#int>
http://lisra.jp/resource/1108	"ラーメン寿がきや名古屋エスカ店"	http://lisra.jp/menu#ML1108a39	"970"^^<http://www.w3.org/2001/XMLSchema#int>
http://lisra.jp/resource/1108	"ラーメン寿がきや名古屋エスカ店"	http://lisra.jp/menu#ML1108a45	"990"^^<http://www.w3.org/2001/XMLSchema#int>
http://lisra.jp/resource/1108	"ラーメン寿がきや名古屋エスカ店"	http://lisra.jp/menu#ML1108a46	"930"^^<http://www.w3.org/2001/XMLSchema#int>
http://lisra.jp/resource/1108	"ラーメン寿がきや名古屋エスカ店"	http://lisra.jp/menu#ML1108a47	"930"^^<http://www.w3.org/2001/XMLSchema#int>
http://lisra.jp/resource/1108	"ラーメン寿がきや名古屋エスカ店"	http://lisra.jp/menu#ML1108a49	"730"^^<http://www.w3.org/2001/XMLSchema#int>
http://lisra.jp/resource/1108	"ラーメン寿がきや名古屋エスカ店"	http://lisra.jp/menu#ML1108a50	"880"^^<http://www.w3.org/2001/XMLSchema#int>
http://lisra.jp/resource/1108	"ラーメン寿がきや名古屋エスカ店"	http://lisra.jp/menu#ML1108a7	"620"^^<http://www.w3.org/2001/XMLSchema#int>
http://lisra.jp/resource/1108	"ラーメン寿がきや名古屋エスカ店"	http://lisra.jp/menu#ML1108a8	"620"^^<http://www.w3.org/2001/XMLSchema#int>
http://lisra.jp/resource/1108	"ラーメン寿がきや名古屋エスカ店"	http://lisra.jp/menu#ML1108a9	"620"^^<http://www.w3.org/2001/XMLSchema#int>

図 17　パターン B 用 SPARQL の結果

表 8　検索結果 (パターン B)

	食べログ	Hot Pepper
適合店舗数	2	2
不適合店舗数	0	3
情報なし	31	
合計	33	5

goal	name
http://lisra.jp/resource/1096	"和食よし川エスカ店"

図 18　評価方法 2 の結果

4.3　考察

自動生成した SPARQL による検索と食べログ・HotPepper による既存のサービスの検索と結果について考察する．表 5，表 6 にある様に，既存のサービスは検索条件が限定されているため，検索クエリ作成の自由度は低い．条件の制限の具体例としては，選定できるエリアが限られているため，メートル単位で指定するような狭い範囲で検索できない，メニューの値段での絞り込み検索ができない等がある．よって，検索ストーリに最適な条件を組み合わせることはできず，時間帯や予算，場所といった要素で詳細な検索ができない．提案プラットフォームのデータ構造では，フィルタタグを DB 内の lisras:FilterProperty をサブクラスに持つプロパティとその値と定義しているので，フィルタタグを要素が追加すれば，システムを変更することなく容易に検索タグの選択肢を拡張することができる．これにより，ユーザのニーズに適した検索が可能である．

情報の順序付けの観点では，食べログは料理スコア，雰囲気スコア，口コミ数等複数の評価軸で順序付けをするこ

とが可能である．Hot Pepper には，ユーザ目線での情報の順序付けという観点はない．各店舗が選択した料金プランに応じて，検索結果が順序付けされている．提案プラットフォームでは，情報の順序付け評価軸であるランクタグを 1 つに限定せず，複数の評価軸で検索を可能する．また，評価軸をユーザスコア，口コミといった形で限定せず，lisras:RankProperty のサブクラスであるプロパティに関連づく値であれば，順序付けを可能とする．そのため，容易に評価軸のカスタマイズが可能である．

各評価結果について考察する．既存のサービスでは，適切な検索クエリを構築することができなかったので，求める情報に絞り込んで得ることはできなかった．また，表 18 にあるように，該当しない店舗だけでなく，店舗名があるだけで十分な情報を提供を行えていない例が多くあった．食べログの情報はユーザが投稿したものであり，Hot Pepper は飲食店側が情報を登録する形式である．この情報収集の形式の違いにより，検索結果に違いが生じてたのだと考える．自動生成した SPARQL に関しては，POI DB には営業時間が各店舗ごとに登録されていることもあり，適切な順序で情報を取得することができた．

考察から得られた必要事項を以下に示す．今回は営業時間で順序付けを行ったが，多様な評価軸で可能とするため，情報の収集・登録を容易に可能とする仕組みの構築．適切な情報取得に限らず，検索ストーリに近い内容で検索した結果をユーザに推薦・掲示する機能これにより，ユーザに新しい発見や気づきといったセレンディピティをもたらし，次の行動の意思決定に影響を与える．

以上のことから，提案する POI のデータ構造に基づいた SPARQL 自動生成手法は，多様なフィルタでの情報の絞り込みや評価軸のカスタマイズが可能であり，DB の拡張にも対応可能であると確認できた．これは本研究で目的とする拡張性やユーザの自由な検索を実現すると考える．

5. まとめ

本稿では，既存の POI 検索サービスにおいてユーザは状況に適した情報を取得できない問題点に対し，ユーザの多様な検索クエリによる自由な検索を実現するため，複数の評価軸で店舗 POI 検索を可能とする店舗 POI 検索プラットフォームを提案した．本プラットフォームの DB の情報の拡張性を高め，柔軟なデータ構造とするため，LOD としての利用を提案し，RDF に対応した POI のデータ構造を提案した．ユーザが検索したい事柄を表現する要素である検索タグという概念を定義した．検索タグの組み合わせから，SPARQL を自動的に生成する手法を提案し，名古屋駅地下街の店舗データを収集し提案プラットフォームの POI DB を実装し評価を行い，自動生成された SPARQL が適切な情報を取得可能であり，DB の拡張に対応可能であると確認できた．

今後の課題を以下に示す．

- ユーザに検索パターンの気付きや新しい発見といったセレンディピティをもたらすため，ユーザ LOD を元に類似度の高いユーザの検索ストーリを推薦する手法の開発
- LOD クラウドを他の LOD とのつながり，データ収集の仕組みの構築．
- 自動的に生成された SPARQL から得られた情報を複数の評価軸で可視化するアプリケーションの開発

謝辞

本研究の一部は，総務省戦略的情報通信研究開発推進事業（SCOPE）132306007 の助成をうけて実施された．

参考文献

[1] 食べログ．http://tabelog.com/ (2014.5.13)
[2] Foursquare．https://ja.foursquare.com/ (2014.5.13)
[3] 飯塚 京士，佐藤 宏之，イコプラムディオノ，村山 隆彦．RDF データを対象としたグラフ検索におけるクエリ生成方式の検討．人工知能学会 SIG-SWO-A502-08, 2005.
[4] 後藤 孝行，濱崎 雅弘，武田 英明．DashSearch LD: 探索的検索の Linked Data への適用．人工知能学会全国大会 (第 26 回)』 3C1-OS-13a-3, 人工知能学会.
[5] Hot Pepper．http://www.hotpepper.jp/ (2014.5.13)
[6] Google Maps．http://maps.google.co.jp/ (2014.5.13)
[7] Yahoo!地図．http://map.yahoo.co.jp/ (2014.5.13)
[8] 大向 一輝. 日本におけるオープンデータの進展と展望.
[9] DBpedia Japanese. http://ja.dbpedia.org/ (2014.5.13)
[10] LODAC. http://lod.ac/ (2014.5.13)
[11] Linked Geo Data．http://linkedgeodata.org/About (2014.5.13)
[12] 中村明順, 通山和裕, 新井イスマイル, 西尾信彦．実世界指向推薦の Coverage 拡大のためのユーザプロファイル抽象化手法. 情報処理学会論文誌, 51(12), pp,.2343-2353, 2010.
[13] 東田 圭介．クラウドソーシングを用いた POI 情報収集．信学技報．AI，人工知能と知識処理 111(447)，pp.17-19，2012-02-21
[14] 鈴木 友基，梶 克彦，河口 信夫．クラウドソーシングによる屋内構造地図情報の構築と収集．信学技報． MoMuC，モバイルマルチメディア通信 111(296)，pp.1-6，2011.
[15] 矢野 幹樹，梶 克彦，河口 信夫．App.Locky：コンテキスト依存型サービス推薦を目的としたユーザ状況収集プラットフォーム．情報処理学会論文誌, 52(12), pp.3274-3288, 2011.
[16] 名駅の地下街．http://www.meieki.com/station_sa.php (2014.5.13)
[17] Jena．http://jena.apache.org/ (2014.5.13)
[18] OpenLink Virtuoso．http://virtuoso.openlinksw.com/dataspace/doc/dav/wiki/Main/ (2014.5.13)