DEIM2022 G33-4

ユーザ定義関数を利用した知識ベースと外部情報源の統合利用手法

大森　雄基†　　北川　博之††　　天笠　俊之†††

† 筑波大学大学院 システム情報工学研究群　〒 305–8577 茨城県つくば市天王台 1 丁目 1–1
†† 筑波大学 国際統合睡眠医科学研究機構　〒 305–8577 茨城県つくば市天王台 1 丁目 1–1
††† 筑波大学 計算科学研究センター　〒 305–8577 茨城県つくば市天王台 1 丁目 1–1
E-mail: †omori.yuki.sm@alumni.tsukuba.ac.jp, ††kitagawa@cs.tsukuba.ac.jp, †††amagasa@cs.tsukuba.ac.jp

あらまし　知識ベースとは，主語・述語・目的語の組の集合である RDF 形式で人間の持つ知識を集積したものであり，様々な知識処理における重要な情報源として活用が進められつつある．しかし，気象情報や運行情報などの即時性が重要な情報や，詳細で専門的な知識など，知識ベース以外の情報源も多く，知識ベースと外部情報源の統合利用により多様な応用に対応することが期待される．一方で，外部情報源の入出力形式は多様であり，その出力結果に知識ベースのエンティティを選出する問題も存在するため，知識ベースと外部情報源の統合利用は一般に極めて煩雑となる．本研究では，SPARQL のユーザ定義関数を利用することで外部情報源を知識ベースと一体として活用できる統合利用環境のアーキテクチャを示す．またその際に重要となる，外部情報源が返す情報に適切な知識ベースのエンティティを選出するエンティティリンキングについても議論する．

キーワード　知識グラフ, SPARQL, エンティティリンキング, データ統合

1　背景と問題

近年，人類の保有する知識の増大ともに，これらの知識を構造化して保持する RDF 形式の知識ベースと，これに対する問合せ言語である SPARQL の利用環境が構築され，様々な知識処理における重要な情報源として活用が進められつつある．しかし，知識の全てが知識ベースに集積されているわけではない．例えば，気象情報や運行情報といった即時性が重要な情報は静的な集積という形式はそぐわない．また，専門的な知識などを提供する知識ベース以外の情報源も多い．このため，これらの情報を有効活用するためには，知識ベースと非知識ベースの外部情報源の連携利用が重要となっている．

しかしながら，外部情報源は，問合せとその返り値に独自の形式を持つことが多く，この形式に対応して情報を解釈する必要がある．このため，知識ベースと外部情報源の連携には，外部情報源の多様な形式に対応しつつ，知識ベースと外部情報源の両方の結果を突き合わせるなど，極めて手間のかかる作業が必要となる．また，外部情報源からの返り値が，知識ベース中のどのエンティティにあたるかは，そのままでは不明であり，これを人手で判別するのも極めて大変な作業となる．このため，知識ベースと外部情報源に対して，**統合的に問合せ**が出来ることが重要となる．

一方，知識ベース検索のに対する問合せ言語 SPARQL では，様々な応用に対応した知識利用のために独自のユーザ定義述語を設定可能 [3] である．これは，SPARQL 問合せ中に応用に応じた計算手続きを組み込むための機能 [4] だが，外部情報へのアクセスにも転用可能である．

本研究では，**ユーザ定義述語を利用して外部情報源にアクセスし，外部情報源をあたかも知識ベースと一体であるかのように扱える統合利用環境のアーキテクチャ**を示す．また，外部情報源からの返り値に対して，知識ベース中の適切なエンティティを選択する手法について述べる．

2　関 連 研 究

2.1　Federation

SPARQL を用いて複数の知識ベースを統合的に検索する技術は Federation と呼ばれ，CostFed [5]、SPLENDID [6] といったものがある．これらは SPARQL 問合せに対して、最適な知識ベースを選択してそれぞれに対応した SPARQL 部分問合せを発行し，その結果を集約することが基本技術である．

本研究では，知識ベースだけではなく非知識ベースの外部情報源を連携させるという問題を扱い，このためのアーキテクチャを設計する．

2.2　エンティティリンキング

エンティティリンキングとは，自然言語中の現実の物事を指し示す語句に対して，知識ベース中の同一の物事を指すエンティティを割り当てる行為である [12]．TAGME [7], DoSeR [8] といった手法がである．

しかし，本研究では自然言語ではなく，外部情報源の返した値を統合するという問題を扱うため，これに合わせてエンティティリンキング法を設計する．

3　前 提 知 識

3.1　RDF 形式の知識ベース

RDF 形式の知識ベースは，主語 s・述語 p・目的語 o のトリプル $s \xrightarrow{p} o$ の集積という形で知識を保存している [1]．主語には現実の物事の IRI (URL, URI の拡張) として表現した**エ**

ンティティが入る．また，目的語にはエンティティに加えて数値・文字列等の値も入る．述語は主語と目的語を関係を指し示す IRI である [13] [1]．あるトリプルの主語や目的語が，別のトリプルの主語や目的語になりうるため，RDF 形式の知識ベースは全体としては図 1 のようにグラフ構造を構成する．このため，知識グラフとも呼ばれる．図 1 では，矩形はエンティティ，文字列は文字列値，エッジは述語を示す．

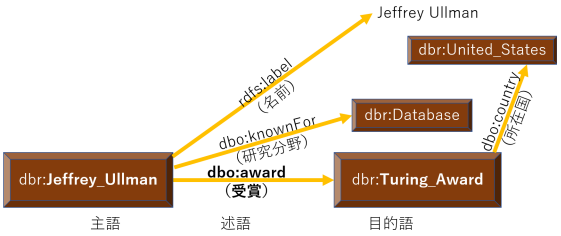


図 1　RDF で作られるグラフの例

3.2　SPARQL

RDF 形式の知識ベースの検索には，問合せ言語 SPARQ が通常用いられる．SPARQL 問合せは，知識グラフの集合に対するパタンマッチングのを行い目的とする情報を取得する [2]．

例えば，以下のような問合せを図 1 に対して実行すると，Jeffrey Ullman を指し示すエンティティが得られる．

```
SELECT DISTINCT * WHERE
{?x rdfs:label “ Jeffrey Ullman"@en . }
```

3.3　ユーザ定義述語

問合せ言語 SPARQL には独自のユーザ定義述語を設定する機能がある [3]．この機能は，知識ベースに対する様々な応用に知識利用する際，SPARQL 問合せ中に応用に応じた計算手続きを組み込むためのもの [4] である例えば図 2 のように，既存のトリプルから生年月日を取得し，現在の年齢を計算するユーザ定義述語を作ることが出来る．

本研究では，この機能を用いて，外部情報源にアクセスして情報を取得し，知識ベースから得られる情報と統合する．

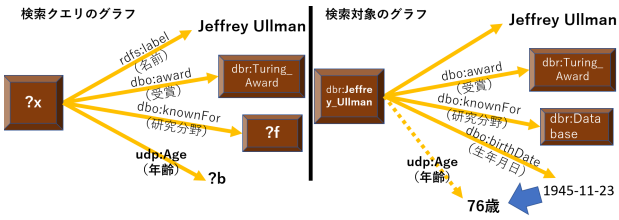


図 2　生年月日から年齢を計算するユーザ定義述語の例

4　提 案 手 法

提案手法のアーキテクチャを図 3 に示す．また，SPARQL 問合せの例を以下に示す．本問合せ例では，<m:co-author> がユーザ定義述語となっている．すなわち，知識ベース中にはこの述語に対応するトリプルは実在せず，主語に対応するエンティティから目的語に対応するエンティティを求めるのは外部情報源を呼び出すことによって行われる．ユーザは，通常の SPARQL 問合せに，アーキテクチャが提供するユーザ定義述語を加えることで，通常の SPARQL で知識ベースを検索したときと同様に知識ベースと外部情報源を統合的に検索可能である．

SPARQL 問合せ例

```
SELECT DISTINCT *
WHERE {
?x rdfs:label "Jeffrey Ullman"@en .
?x dbo:award dbr:Turing_Award .
?x dbo:knownFor ?f .
?x <m:co-author> ?y.
}
```

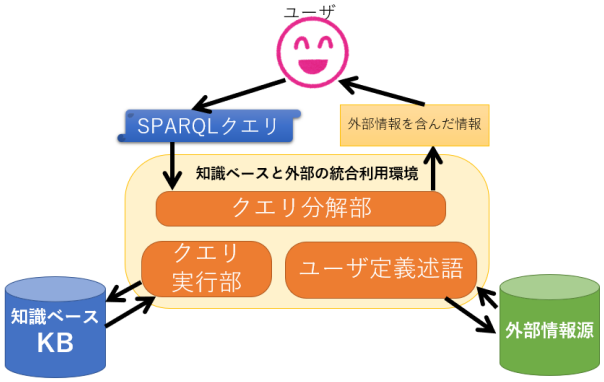


図 3　提案手法のアーキテクチャ

4.1　提案アーキテクチャ

提案手法のアーキテクチャ（図 3）は，SPARQL 問合せを再構成する機能，再構成した SPARQL 問合せ等を利用して知識ベース中を検索する機能，ユーザ定義述語に対応して外部情報源にアクセスする機能から成る．

4.2　提案アーキテクチャがユーザに提供する拡張知識グラフ

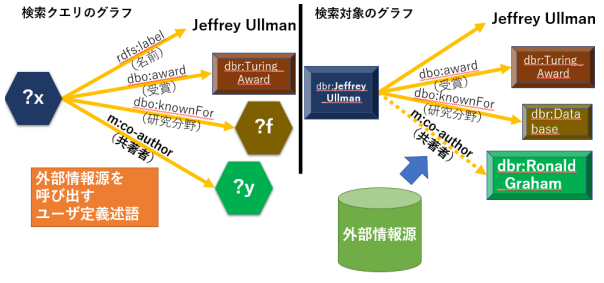


図 4　提案手法のアーキテクチャが取り扱う問合せを示すグラフと，対応する知識グラフ．

図 4 の右側は，提案手法のアーキテクチャが知識グラフを拡張することを示した図である．知識ベース dbpedia 中では，dbr:Jeffrey_Ullman というエンティティは名前を示す

述語 rdfs:label を通じて Jeffrey Ullman という文字列を持っている．また，述語 dbo:knownFor によって エンティティ dbr:Database と接続されることにより，研究分野としてデータベースがあることを示す．同様に，述語 dbo:award と エンティティdbr:Turing_Award により，チューリング賞受賞者であることがわかる．提案手法では，元の知識グラフには存在しないユーザ定義述語 m:co-author を加え，更に元の知識グラフに無い $\texttt{dbr:Jeffrey\_Ullman} \xrightarrow{\texttt{m:co-author}} \texttt{dbr:Ronald\_Graham}$ というトリプルを存在するとすることにより，知識グラフの拡張を行う．

4.3 ユーザ定義述語を用いるために事前にやるべきこと

ユーザが提案手法のアーキテクチャを利用するには，以下 2 つの事項をアーキテクチャに登録する．(1) ユーザ定義述語の IRI．(2) エンティティx_i に対応する検索値 $o(x_i)$ が与えられたときの外部情報源へのアクセス方法 $M_i = f_o(x_i)$．ただし，M_i はエンティティに対応する取得値の集合である．

また，エンティティx から検索値への変換 $o(x_i)$ は，x_i の名称 (rdfs:label) 以外の情報が必須であるときに定める必要がある．

4.4 拡張知識グラフに対する SPARQL 問合せ

図 4 の左側は，SPARQL 問合せ例をグラフとして示したものである．?x <m:co-author> ?y．の一行を除いて，問合せ例は通常の SPARQL 問合せである．

提案手法のアーキテクチャに問合せを行うとき，ユーザはユーザ定義述語の IRI を述語として通常の SPARQL 問合せに加える．また，ユーザ定義述語の目的語は，変数である必要がある．ユーザ定義述語の主語は，エンティティの IRI であるか，他の部分で出現した変数である必要がある．(主語に制約のない変数を使用することも理論上は考えうるが，この場合は知識ベース中に存在するエンティティの個数ぶんだけ外部情報源に問合せる，あるいは外部情報源のエンティティに相当の個数分の情報を問合せるといった状況となるため，使用状況に現実性があるとは考えられないため，本研究の対象とはしない.)

4.5 拡張 SPARQL 問合せの処理手順

提案アーキテクチャにおける問合せ処理の仕組みは以下のステップで構成される．

1. ユーザ定義述語以外の部分について、SPARQL 問合せを実行
2. ユーザ定義述語に関する外部情報源から情報を取得
3. エンティティの割り当て

3-1 ユーザ定義述語の主語に当たるエンティティに関連する情報を追加取得する．

3-2 Step 2 で得た外部情報源の結果から、エンティティと語句の対応辞書より、候補エンティティを選出する．

3-3 関連度スコアを用いて、候補エンティティの中から最適なエンティティを決定する。もしくは、最適なエンティティがないと宣言する．

4 最終的な問合せ結果の導出

Step 1：ユーザ定義述語以外の問合せを実行

Step1 では、ユーザが送った問合せからユーザ定義述語を含む部分を削除したうえで，知識ベースに対するマッチングを行う．これは通常の SPARQL 問合せとなるため，知識ベースからはタプルの列が返される．これを R_Q とする．

また、R_Q 中に出現したユーザ定義述語の主語に当たるエンティティの集合を X とする．この要素を主語エンティティと呼び，$x_i \in X$ と表記する，また、x_i と同じタプルに出現するエンティティの集合を Q_i とする．

以上によって，ユーザ定義述語に対する主語のにあたるエンティティと、それに関連する情報が得られる．

Step 2：外部情報源からの情報取得

Step 2 では、サーバ定義述語内の関数で、外部情報源へのアクセスを行う．Step 1 で得たユーザ述語関数の主語にあたるエンティティ$x_i \in X$ について，それを外部情報源インタフェイスに適した形式への変換を $o(x)$ とする．これを用いて，外部情報源にアクセスしたとき，主語エンティティx_i に対応する返り値の集合を M_i とする．

Step 3：エンティティの割り当て

Step 3 では，Step 2 で得た外部情報源の検索結果について，知識ベース中のどのエンティティに対応するか，もしくは，エンティティが無いと判定する．

各サブステップについては次に述べる．

Step 3-1：文脈エンティティの抽出

ユーザ述語関数の主語にあたるエンティティ$x^i \in X$ について，パラメータ N に従い，その関連するエンティティを取得する．

$n = 1$ のとき，知識ベースからユーザ定義述語の主語 $?x$ が，主語になっているトリプル $?x \xrightarrow{?p1} ?e_1$ を取得し，その目的語にあたるエンティティ$?e_1$ の集合 E_1^i を追加取得する．ただし，x_i と Q_i に含まれるエンティティは除外する．

$n = r$ のとき，知識ベースから $?e_{r-1}$ が主語となっているトリプル $?e_{r-1} \xrightarrow{?p_r} ?e_r$ を取得し，その目的語にあたるエンティティ$?r_n$ の集合 E_r^i を再帰的に追加取得する．ただし，x_i と Q_i に含まれるエンティティ，および $E_1^i \ldots E_{r-1}^i$ に含まれるエンティティは除外する．

これを $n = N$ となるまで繰り返す．

x_i 自身と Q_i，以上で集めたエンティティの集合 $E_1^i \ldots E_N^i$ をあわせて文脈エンティティ集合 $ctx(x_i, N)$ と呼ぶ．すなわち，$ctx(x_i, N) = \{x_i\} \cup Q_i \cup \bigcup_{n=1}^{N} E_n^i$ となる．

なお，$N = 0$ のときは，$E_1^i \ldots E_N^i$ は利用しない．また，$N = -1$ のときは Q_i も利用せず x_i 自身のみを含む集合となる．

Step 3-2：候補エンティティの選出

候補エンティティの選出とは，Step 2 で得た外部情報源の結果 $m_{ij} \in M_i$ に対して，同一の事物を指している可能性のある知識ベース中の候補エンティティの集合 C_{ij} を選び出すこと

である．あらかじめ用意した語句と候補エンティティとの対応辞書を用いて行う．

この辞書の構築には，英語版 Wikipedia から収集したアンカーリンク情報を利用する．アンカーリンクは，Wikipedia 中の語句から Wikipedia ページへ飛ぶリンクであり，語句と Wikipedia ページの対応関係が得られる．これを対応するエンティティへ変換する．

Step 3-3：最適なエンティティの決定

このステップのでは，m_{ij} に対して，適切なエンティティ $c_{ijk} \in C_{ij}$ を選択する，もしくは知識ベース中には無いと判定することである．これを行うため，x_i と c_{ijk} の関係を関連度スコア $score(x_i, c_{ijk})$ として計算し，最も x_i と関連するエンティティを C_{ij} の中から判定する．関連度スコア $score(x_i, c_{ijk})$ の計算方法として，(3-3-a)TagMe 方式，(3-3-b)Jaccard 方式，(3-3-c)Word Vector 方式の 3 つを考える．

最適エンティティは，

$$y_{ij} = \begin{cases} \arg\max_k score(x_i, c_{ijk}) & max(score(x_i, c_{ijk})) > th \\ Null & othewize \end{cases}$$

となるエンティティ y_{ij} である．ただし，th は閾値である．

以上から，ユーザ定義述語の主語に当たる x_i と，目的語に当たるエンティティ y_{ij} が取得できたので，これをタプルとしたものの集合が返すべき値である．これを R_{UDP} とする．

Step 3-3-a：TagMe 方式 [7]

エンティティの関連度 $rel(e_a, e_b)$ は，エンティティ e_a に相当する Wikipedia ページを w_a，e_b に相当する Wikipedia ページを w_b，w_a のアンカーリンクが指し示すページの集合を W_l^a，w_b からのものを W_i^b とすると，$rel(e_a, e_b) = |W_l^a \cap W_l^b|$ である．つまり，w_a, w_b の指し示すページが重複している数である．

関連度スコアは，候補エンティティ c_{ijk} とそれぞれの文脈エンティティとの関連度の合計とする．

$$score(x_i, c_{ijk}) = \sum_{e_{ctx} \in ctx(x_i, N)} rel(c_{ijk}, e_{ctx}).$$

Step 3-3-b：Jaccard 方式

Jaccard 方式ではエンティティの概要文書を利用する．すなわち，エンティティ e に対して，$e \xrightarrow{dbo:abstract} a$ を用いてその概要を示すテキスト a を取得する．エンティティ e の概要テキスト a から単語の集合を取り出す操作を $A(e)$ とする．個々のエンティティの関連度 $rel(e_1, e_2)$ は，e_1 と e_2 の概要テキストを利用し，Jaccard 係数を計算する．

$$rel(e_1, e_2) = \frac{|A(e_1) \cap A(e_2)|}{|A(e_1) \cup A(e_2)|}$$

関連度スコアは，候補エンティティ c_{ijk} とそれぞれの文脈エンティティとの関連度の平均とする．

$$score(x_i, c_{ijk}) = \frac{1}{|ctx(x_i, N)|} \sum_{e_{ctx} \in ctx(x_i, N)} rel(c_{ijk}, e_{ctx}).$$

Step 3-3-c：Word Vector 方式

Jaccard 方式と同様に概要文書を利用するが，予め計算された単語の埋込ベクトルを利用するという点で異なる．

単語埋込ベクトルは，Wikipedia2Vec [9] を用い，エンティティ e のベクトル $vec(e)$ は概要テキスト中の単語埋込ベクトルの平均を用いる．エンティティの関連度 $rel(e_1, e_2)$ は，e_1 と e_2 のベクトルの cos 類似度とする．

$$rel(e_1, e_2) = \cos(vec(e_1), vec(e_2))$$

関連度スコアは，候補エンティティ c_{ijk} とそれぞれの文脈エンティティとの関連度の平均とする．

$$score(x_i, c_{ijk}) = \frac{1}{|ctx(x_i, N)|} \sum_{e_{ctx} \in ctx(x_i, N)} rel(c_{ijk}, e_{ctx}).$$

Step 4：最終的な問合せ結果の導出

最後のステップでは、Step1 の問合せ結果と、Step3 の最適エンティティを利用して、最終的な問合せ結果をユーザに返す．すなわち，Step 1 で取得したユーザ定義述語を取り除いた問合せの実行結果である R_Q と，Step 3 で計算したユーザ定義述語に対する結果 R_{UDP} を用い，$R_Q \bowtie R_{UDP}$ が最終的に返す値である．ただし，$\bowtie$ とは，リレーショナルデータベースにおける自然結合である．

5 実　　験

5.1 実 験 環 境

実験には，知識ベースには英語版 DBpedia を用いる．これは英語版 Wikipedia を RDF 形式に変換したものである [10]．また，外部情報源には論文情報データベース DBLP を用いる．Grand Truth は存在しないため，目視確認で正解データを作成する．

ユーザ定義述語は，DBLP からダウンロードしたデータを用い，共著者情報を返すものである．ユーザ定義述語の主語に当たるエンティティ $?x$ について，その名前を示す述語 `rdfs:label`, `foaf:name`, `dbp:name` を用いて名前 $o(?x)$ を取得する．この名前から論文を用いて DBLP から該当論文とその著者情報を取得し，共著者の名前とする．あとは，Step 1,2,3,4 で示した通り，共著者のエンティティリンキングを行い，それぞれの主語エンティティ $?x$ に対して，目的語エンティティ $?y$ に当たるエンティティの集合を提示する．

この環境のもと，以下の 4 つの問合せを実行し，リンキング結果を検証した．なお，これらの問合せは通常の SPARQL 問合せ処理のベンチマーク [11] に利用されたものを参考に作成した．

(Query1) 知識ベース中のある研究者を指定し，その共著者を取得する問合せ．（正解データ数 28 件）

```
SELECT DISTINCT * WHERE {
?x rdfs:label "Dan Hirschberg"@en .
?x rdf:type dbo:Scientist .
?x dbo:knownFor ?k .
?x dbp:workplaces ?w .
?w dbo:city ?c .
?x <m:co-author> ?y. }
```

(Query2) 特定分野の研究者を知識ベース中から探し，共著者を取得する問合せ．（正解データ数 35 件）

```
SELECT DISTINCT * WHERE {
?x dbo:knownFor dbr:Combinatorial_design .
?x dbo:nationality ?n .
?x dbo:residence ?r .
?x <m:co-author> ?y.}
```

(Query3) 非情報系の指定した本の著者を知識ベース中から探し，共著者を取得する問合せ．（正解データ数 12 件）

```
SELECT DISTINCT * WHERE {
?book dct:subject dbc:Linguistics_books .
?book dbp:name ?name .
?book dbp:pages ?pages .
?book dbp:isbn ?isbn .
?book dbp:author ?x .
?x rdf:type dbo:Scientist .
?x <m:co-author> ?y. }
```

(Query4) 茨城県の大学に所属する研究者を知識ベース中から探し，共著者を取得する問合せ．（正解データ数 61 件）

```
SELECT DISTINCT * WHERE {
?x rdf:type dbo:Scientist .
?x dbo:institution ?univ .
?univ dbo:state dbr:Ibaraki_Prefecture .
?x <m:co-author> ?y. }
```

5.2 実験結果

実験結果の評価は，ユーザ定義述語の主語とリンキング結果である目的語の組の集合について，リンキング結果と真の出力を比較したものを集計した．なお，Precision が空欄の部分は，統合環境が，全ての $m_{ij} \in M_i$ について，エンティティが無いと判定したことを示す．

5.2.1 TagMe 方式のパラメータ

図 5 は，TagMe 方式の Query1, 2, 3, 4 に対する結果である．

Query1 では，$(N, th) = (2,0), (1,0)$ のとき，Precision, Accuracy が高い一方，Recall は，$(N, th) = (3,0)$ のときが最も高い．

Query2 では，$(N, th) = (3,0)$ のときが最も性能が高い．

Query4 では，$(N, th) = (2,5)$ のとき，Precision が高いが Recall は極めて低い．$(N, th) = (3,0)$ の時が Recall が最も高く，$(N, th) = (3,5)$ のとき Accuracy が高く，Precision も二番目に高い．

また，Query1, 2, 4 では，N が小さいときは正解のエンティティを探索する能力は小さい．一方，Query3 では，正解エンティティを探し出せなかった．

図 6 は，以上の結果から各クエリ Precision・Recall のマクロ平均と，F 値を示したものである．$(N, th) = (3,0)$ のとき，F 値，Recall が最大であり，$(N, th) = (3,5)$ のとき，Precision が最大で F 値が 2 番目であるため，TagMe 方式は $(N, th) = (3,0), (3,5)$ の場合を取り扱う．

5.2.2 Jaccard 方式のパラメータ

図 7 は，Jaccard 方式の Query1, 2, 3, 4 に対する結果である，Jaccard 方式では明確に他より優れたパラメータが存在す

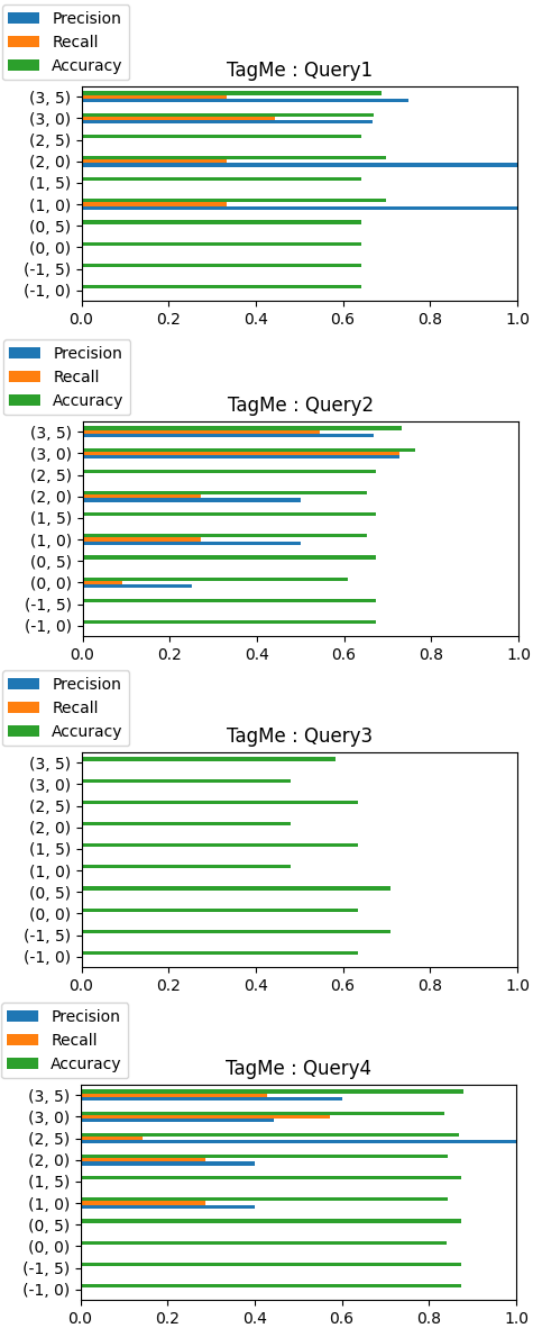


図 5　リンキング結果：TagMe 方式

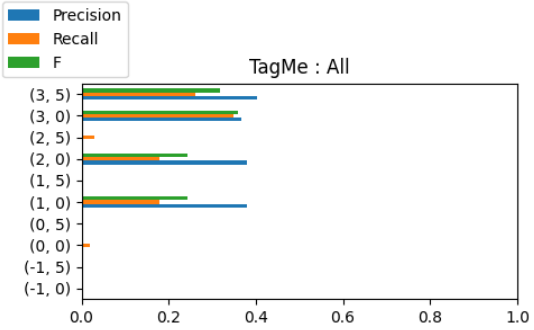


図 6　TagMe 方式の全クエリの平均値と F 値

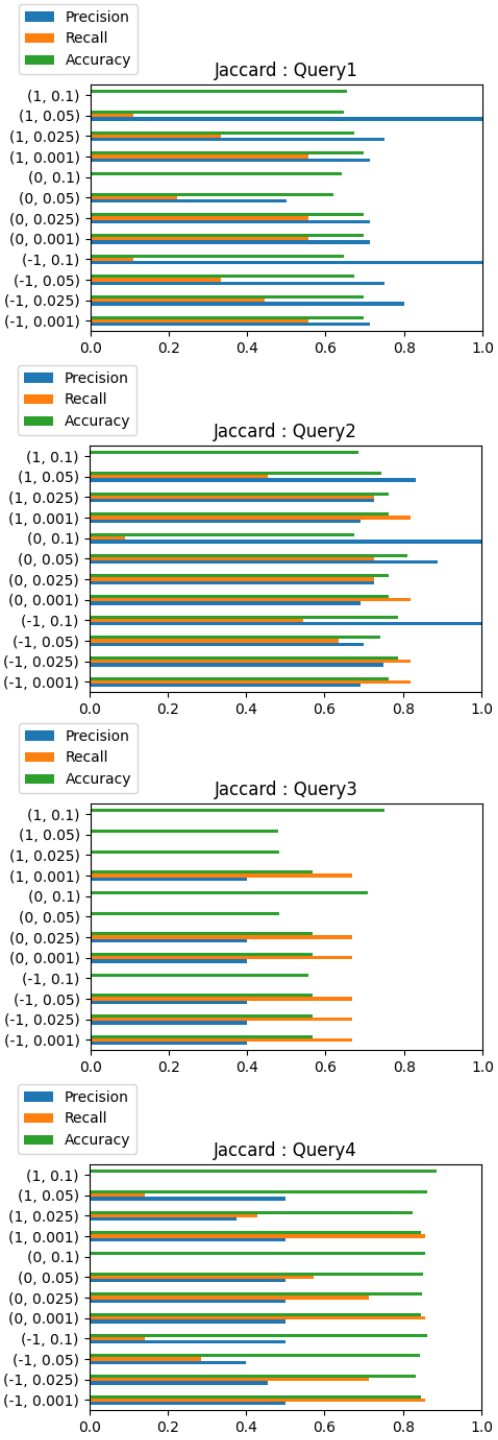


図 7　リンキング結果：Jaccard 方式：Query1,2

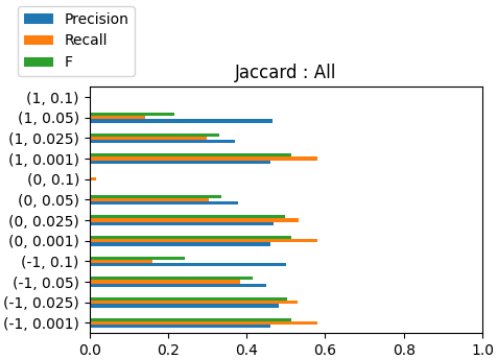
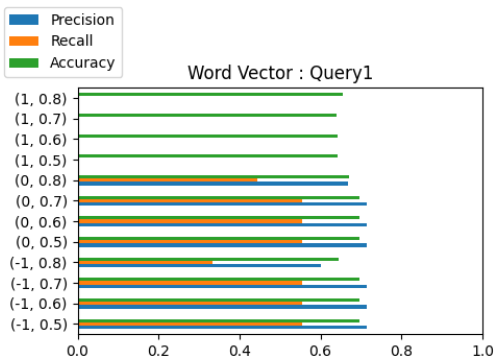
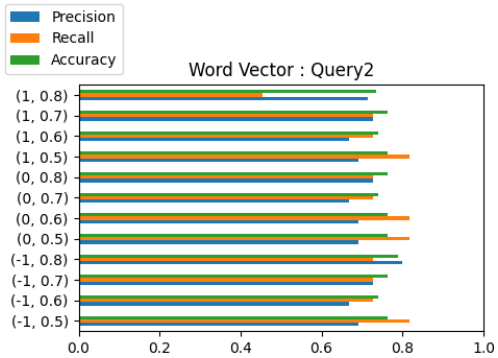
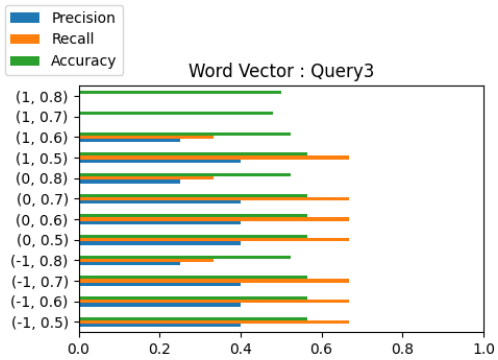


図 8　Jaccard 方式の全クエリの平均値と F 値







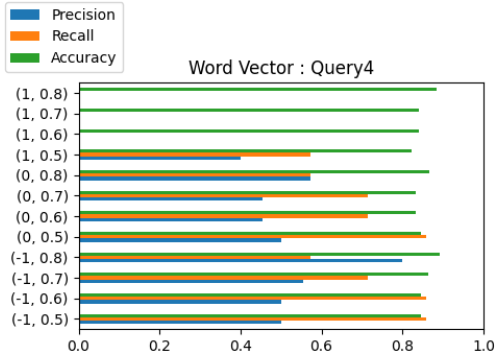


図 9　リンキング結果：Word Vector 方式

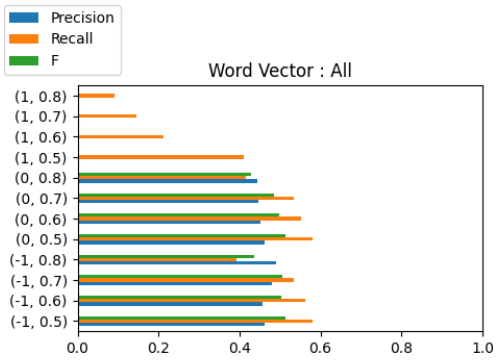


図 10　Word Vector 方式の全クエリの平均値と F 値

るわけではない．よって，今後は最もシンプルな方式である $N=-1$ について議論する．$N=-1$ のとき，Query1, 4 では，$th=0.001$ のとき Recall が高く，Query2, 3 では $th=0.001$ も $th=0.005$ も変わらない．

図 8 は，以上の結果から各クエリ Precision・Recall のマクロ平均と，F 値を示したものである．$(N,th)=(-1,0.001),(-1,0.025)$ の 2 つが優れるが Precision の差はあまりないため，Recall と F 値の大きい $(N,th)=(-1,0.001)$ の場合を取り扱う．

5.2.3 Word Vector 方式のパラメータ

図 9 は，TagMe 方式の Query1, 2, 3, 4 に対する結果である．Query2 の (N,th)=(0,0.6) 以外において，同じ th では N が小さいほうが同等であるか優れているため，$N=-1$ と $(N,th)=(0,0.6)$ のときについて検討する．

Query1, 3 では，$N=-1,0, th=0.6,0.7$ のとき最も性能が高い．

Query2 では，$(N,th)=(-1,0.8),(-1,0.9)$ のとき，Precision, Accuracy が高い一方，$(N,th)=(-1,0.5),(0,0.6)$ のとき，Recall が高い．

Query4 では，$(N,th)=(-1,0.8)$ のとき，Precision, Accuracy が高い一方，$(N,th)=(-1,0.5)$ のとき，Recall が高い．

図 10 は，以上の結果から各クエリ Precision・Recall のマクロ平均と，F 値を示したものである．以上より，Word Vector 方式では Recall と F 値の大きい $(N,th)=(-1,0.5)$ と，Precision の大きい $(N,th)=(-1,0.8)$ の場合を取り扱う．

5.2.4 各手法の比較

図 11 は，Query1, 2, 3, 4 について，これまでの結果からパラメータを定めて各手法を比較したものである．

全体的な傾向として，Accuracy にはあまり差が無い．

Recall においては Jaccard 方式の $(N,th)=(-1,0.001)$ と Word Vector 方式の $(N,th)=(-1,0.5)$ が高い．

Precision は， Query1 では TagMe 方式の $(N,th)=(3,5)$ が優れるが，Query2, 4 では Word Vector 方式の $(N,th)=(-1,0.8)$ が優れ，Query3 では Jaccard 方式の $(N,th)=(-1,0.001)$ と Word Vector 方式の $(N,th)=(-1,0.5)$ が優れる．

5.3 考　　察

どの方式でも，知識ベース中に存在しないエンティティの却下の能力は高い．

また，TagMe 方式においては，適切な推定には文脈エンティティが多く $(N=3)$ 必要であることがわかった．エンティティ間のグラフ関係が豊富な場合は正しいエンティティを判定でき，一方で，エンティティ間のグラフ関係が疎な場合に，誤却下や誤ったエンティティに誘導される事例があった．

一方，Jaccard 方式や Word Vector 方式では，文脈エンティティを増加しても推定の性能に寄与しない．

Jaccard 方式や Word Vector 方式が TagMe 方式よりも性能が高い場合が多く，グラフ関係を利用するより，エンティティ

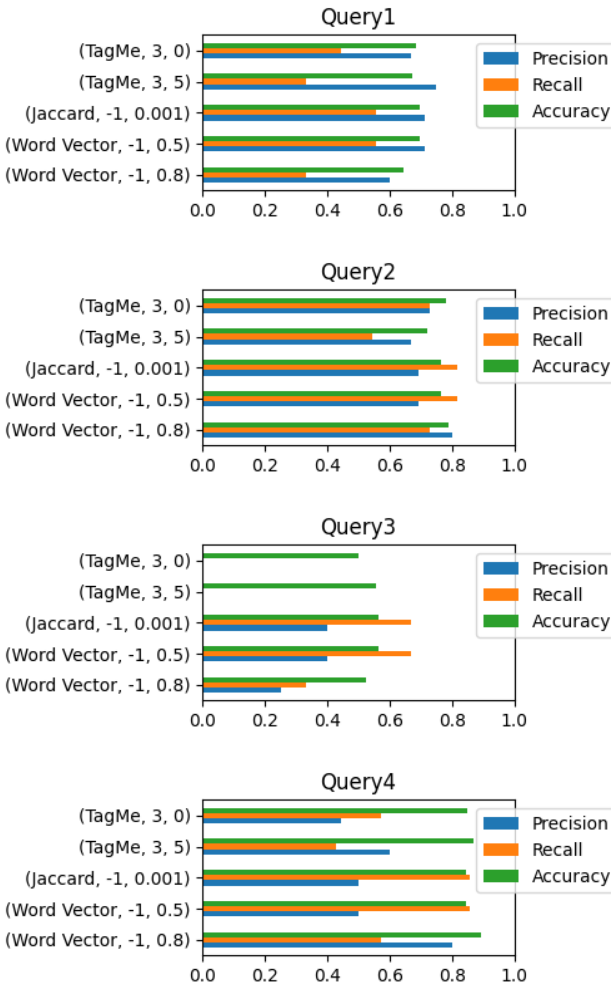


図 11　問合せ別のリンキング結果比較

e の概要文書 $A(e)$ を利用するほうが優れていると言える．

4 つの問合せのなかで，最も顕著な差が出たのは Query3 である．Query3 では TagMe 方式は全くエンティティを探し出せなかった．Query3 は情報系の論文が多い外部情報源を利用し，言語学との関連を見る問合せである．つまり，研究分野等から考えると，他の問合せに比べてエンティティ間の関係がグラフとして繋がっていない可能性が高い問合せと言えるため，グラフ関係から判定する TagMe 方式は機能しなかったと考えられる．一方，概要テキストを利用する Jaccard 方式や Word Vector 方式では，アンカーリンクの付与されていない語句（つまり dbpedia 上のグラフ関係に反映されていない関係）の近接性を測ることが出来たと言える．

また，Query4 でも，エンティティを探し出す能力が Jaccard 方式や Word Vector 方式に比べて低い．Query4 では日本の研究者のエンティティが多いため，英語版 dbpedia では関わるトリプルが少ないことが多い．よって，これも Query1 や Quey2 に比べてエンティティ間の関係をグラフでとらえにくい問合せであると言える．

以上のことから，グラフ関係を利用するより，概要テキスト $A(e)$ を利用する方式が優れていると言える．しかし，両者の結果を比較すると，必ずしもそうは言えない．表 1 は，Query1

から Query4 の 28+35+12+61=136 件の組について，TagMe 方式 $(N, th) = (3, 5)$ と Jaccard 方式 $(N, th) = (-1, 0.001)$ のリンキング結果を表としたものである．TagMe 方式では知識ベース中で存在するにも関わらず無いと判定した 8 件について，Jaccard 方式では正しいエンティティを導出した．また，Jaccard 方式で誤って存在すると判定した 7 件について，TagMe 方式では正しく存在しないと判定している．また，表 2 より TagMe 方式 $(N, th) = (3, 5)$ と Word Vector 方式 $(N, th) = (-1, 0.5)$ でもほぼ同じである．このことから，グラフ関係を使った方式とエンティティ e の概要テキスト $A(e)$ を利用する方式は，ある程度補完しあう関係があるために，これらを組み合わせることに意味があると考えられる．

表 1　リンキング結果の比較．左側が TagMe 方式（$N = 3$, 閾値 5）のリンキング結果を，上側が Jaccard 方式（$N = -1$, 閾値 0.001）のリンキング結果を示す．「=」はリンキングの判定と Grand Truth が等しい場合を示す．「≠」は Grand Truth に該当エンティティがあってもリンキングでは別のエンティティと判定した場合を示す．「FN」は，Grand Truth に該当エンティティがあってもリンキングでは無しと判定した場合を示す．「FP」は，Grand Truth に該当エンティティが無くてもリンキングでは存在すると判定した場合を示す．「TN」は，Grand Truth に該当エンティティが無いとき，リンキングでは正しく無しと判定した場合を示す．

		Jaccard 方式 $(N, th) = (-1, 0.001)$				
		=	≠	FN	FP	TN
TagMe 方式 $(N, th) = (3, 5)$	=	11	0	0	0	0
	≠	1	0	0	0	0
	FN	8	1	9	0	0
	FP	0	0	0	8	0
	TN	0	0	0	7	91

表 2　リンキング結果の比較．左側が TagMe 方式（$N = 3$, 閾値 5）のリンキング結果を，上側が Word Vector 方式（$N = -1$, 閾値 0.5）のリンキング結果を示す．行名・列名は表 1 と同じである．

		Word Vector 方式 $(N, th) = (-1, 0.5)$				
		=	≠	FN	FP	TN
TagMe 方式 $(N, th) = (3, 5)$	=	11	0	0	0	0
	≠	1	0	0	0	0
	FN	9	1	8	0	0
	FP	0	0	0	8	0
	TN	0	0	0	7	91

6 ま と め

本研究では、知識ベースと非知識ベース外部情報源の統合利用アーキテクチャを提案した．また、この問題に合わせたエンティティリンキングを設計した．また，複数のリンキング方式を比較することで，より高度なリンキングの可能性を示した．今後の課題は，より多様な UDP を用いた検証．より高度なエンティティリンキングを設計すること．また、ユーザ定義述語が複数ある場合の問合せ処理方法を定めることである．

7 謝　　辞

本研究の一部は，JSPS 科研費 JP19H04114，AMED ムーンショット型研究開発事業による．

文　　献

[1] RDF 1.1 Concepts and Abstract Syntax, W3C Recommendation 25 February 2014 , https://www.w3.org/TR/rdf11-concepts/
[2] SPARQL 1.1 Query Language, W3C Recommendation 21 March 2013 , https://www.w3.org/TR/2013/REC-sparql11-query-20130321/
[3] 3.3 Magic Properties, SPIN - Modeling Vocabulary, W3C Member Submission 22 February 2011 https://www.w3.org/Submission/spin-modeling/#spin-magic
[4] Feature:JavaScriptFunctions, W3C, https://www.w3.org/2009/sparql/wiki/Feature:JavaScriptFunctions
[5] Muhammad Saleem, et al., "CostFed: Cost-Based Query Optimization for SPARQL Endpoint Federation", Procedia Computer Science, Volume 137, 2018.
[6] Olaf Görlitz and Steffen Staab, "SPLENDID: SPARQL endpoint federation exploiting VOID descriptions. In Proceedings of the Second International Conference on Consuming Linked Data" COLD'11, 2011.
[7] Paolo Ferragina and Ugo Scaiella, "TAGME: on-the-fly annotation of short text fragments (by wikipedia entities)", CIKM, 2010.
[8] Stefan Zwicklbauer, et al., "DoSeR - A Knowledge-Base-Agnostic Framework for Entity Disambiguation Using Semantic Embeddings." ESWC, 2016.
[9] Ikuya Yamada, et al., "Joint Learning of the Embedding of Words and Entities for Named Entity Disambiguation" ACL, 2016.
[10] Jens Lehmann, et al., "DBpedia – A large-scale, multilingual knowledge base extracted from Wikipedia" Semantic Web, vol. 6, 2015.
[11] Mohamed, et al., "DBpedia SPARQL Benchmark – Performance Assessment with Real Queries on Real Data" ISWC, 2011.
[12] Özge Sevgili, et al., "Neural Entity Linking: A Survey of Models Based on Deep Learning" SWJ, 2021.
[13] Farzaneh Mahdisoltani, et al., "YAGO3: A Knowledge Base from Multilingual Wikipedias" CIDR, 2015.