DEIM2022 G33-3

複数の知識ベースに対するキーワード検索

中野茉里香† 天笠 俊之††

† 筑波大学 情報理工学位プログラム 〒 305–8573 茨城県つくば市天王台 1–1–1
†† 筑波大学 計算科学研究センター 〒 305–8573 茨城県つくば市天王台 1–1–1
E-mail: †marikan@kde.cs.tsukuba.ac.jp, ††amagasa@cs.tsukuba.ac.jp

あらまし 近年，事物に関する一般的な知識を構造化データとして表現し，機械処理に利用可能な知識ベースが注目され，様々な分野で利用されている．SPARQL 言語を用いることで知識ベースの情報への問合せが可能となるが，ユーザは SPARQL や知識ベースのスキーマに精通している必要がある．これに対して，キーワードクエリから知識ベースに問合せを行うことでリソースを獲得する手法が提案されてきたが，知識ベースが単一である場合に限定され，専門知識のないユーザが複数の知識ベースの中から問合せを行うデータセットを選択する必要があった．さらに，知識ベースはそれぞれ異なるターゲットやドメインを持ち，単一の知識ベースに対する問合せのみでは得られない情報が存在する．そこで本研究では，複数知識ベースへのキーワードマッチングと，マッチングされたリソースを含む異種知識ベース間の最小サブグラフの抽出を行うことで，横断的なキーワード検索を行う手法を提案する．

キーワード 知識ベース，セマンティック Web，情報統合

1 はじめに

近年，事物に関する一般的な知識を構造化データとして表現し，機械処理に利用可能な知識ベースが注目され，様々な分野で利用されている．一般的な知識ベースでは，ファクトは主語-述語-目的後からなるトリプルとして記述され，あらゆる知識はトリプルの集合として表現される．RDF(Resource Description Framework) [1] はその主要なモデルの１つである．これによって，既存の知識ベースは数十億のエンティティとその関係を知識グラフとして網羅している．

知識ベースへの問合せは SPARQL(SPARQL Protocol and RDF Query Language) [2] を標準言語とし，基本グラフパターン (BGP) により問合せ条件を指定する．さらに SPARQL は分散した知識ベースへの問合せを実現する横断問合せをサポートする．

現在，知識ベースはターゲットやドメインの異なる 60 万以上のデータセットが存在するとされている [3]．例えば，DBPedia[1] [4],YAGO[2], Wikidata[3]は，いずれも知識を幅広く網羅する代表的な汎用大規模知識ベースとして知られているが，保有する情報の粒度や語彙体系に多くの違いがある．さらに，地理情報に特化した知識ベースである GeoNames [5]，薬物に関する情報を包括的に持つ知識ベースである DrugBank [6]，化合種を対象とした知識ベースである ChEBI [7] など，特定のドメインに特化した知識ベースも存在する．このため，目的の情報に関するデータを広く取得するためには，単一知識ベースに限定せず，複数知識ベースを対象に問合せを行う必要がある．

そこで，複数知識ベースを横断した問合せを行う手法が提案されている [8], [9]．これらの手法では，入力クエリについて，その BGP をサブクエリに分解することで，単一クエリからの複数知識ベースへの問合せを実現している．各サブクエリについて，結果を持つ可能性が高いと判断された知識ベースと結び付けられて，問合せが実行される．全てのサブクエリが処理されると，部分的な結果が結合されて最終的な結果が生成される．

また，知識ベースの情報への問合せは SPARQL を用いることで可能となるが，ユーザは SPARQL 構文や知識ベースのスキーマに精通している必要がある．さらに，知識ベースには膨大な量の背景知識が存在するため，熟練者でないユーザだけでなく，専門家にとっても SPARQL クエリを作成することが困難となる場合がある．

これに対して，キーワードクエリから知識ベースに問合せを行うことで，より直感的にリソースを獲得する手法 [10] や [11] が提案されてきたが，その多くが知識ベースが単一である場合に限定され，専門知識のないユーザが複数の知識ベースの中から問合せを行うデータセットを選択する必要があった．さらに，知識ベースはそれぞれ異なるターゲットやドメインを持ち，単一の知識ベースに対する問合せのみでは得られない情報が存在するが，キーワード検索ではこれらを横断するリソースに対する問合せが実現されていない．

そこで本研究では，複数知識ベースへのキーワードマッチングと，マッチングされたリソースを含む異種知識ベース間の最小サブグラフの抽出と結合を行うことで，横断的なキーワード検索を行う手法を提案する．

本稿の構成は以下の通りである．

まず，２節で本研究における前提知識について説明する．３節では関連研究について説明する．４節では提案手法の説明を行い，５節では評価実験について述べる．最後に６節で本研究の結論と今後の方針についてまとめる．

1：http://dbpedia.org/
2：https://yago-knowledge.org//
3：http://wikidata.org/

2 前提知識

2.1 知識ベース

知識ベースとは様々な分野の知識を組織化し，蓄積したデータベースである．大規模な知識ベースの代表例としてはWikipedia 等がある．知識ベースの中でも特に，構造化され，機械処理に利用可能な知識ベースが増えている．このような知識ベースは RDF (Resource Description Framework)（後述）のようなデータ形式を用いて論理的に一貫した形式で記述がなされている．構造化された知識ベースの代表例としては，DBPedia[4] [4],YAGO[5], Wikidata[6]などが存在する．

2.2 RDF (Resource Description Framework)

知識ベース等の様々な資源を構造的に記述するためのデータ形式として，RDF(Resource Description Framework) [1] がある．RDF とは，Web 上のリソースのメタデータを記述するための枠組みである．

RDF データは最小単位として，主語 (Subject)-述語 (Predicate)-目的語 (Object) からなるトリプルを持つ．トリプルの各要素は，基本的には Web 上のデータの識別子である URI(Uniform Resource Identier) を用いて表現される．

URI 以外の表現方法としては主語，述語，目的語の各要素ごとに制約が異なる．主語は URI または空白ノードで構成される．述語は URI を持つ．目的語は URI またはリテラルか空白ノードで構成される．リテラルとは，URI のように形式化された識別子ではなく文字列をそのまま記述したものを示す．

この RDF トリプルを組み合わせることによって RDF グラフが構成される．RDF グラフは，主語と目的語がノード，述語がエッジを示すラベル付き有向グラフとして表現される．

2.3 SPARQL (SPARQL Protocol and RDF Query Language)

知識ベースから目的のデータを取得するために問合せを行う言語が，SPARQL(SPARQL Protocol and RDF Query Language) [2] である．ユーザは SPARQL クエリで特定のグラフパターンを指定することで，照合するグラフパターンを Web 上から検索することができる．

3 関連研究

3.1 複数知識ベースへの横断問合せ

対象とするドメインや汎用性，サイズなどの様々な局面が異なる知識ベースが非常に増加している．DBpedia や Freebase のような汎用的な知識ベースは幅広い事実に関する一般的な知識ベースを扱い，GeoNames のような分野固有の知識ベースは対象ドメインにより特化した知識を扱う．このため，2 つ以上の知識ベースを組み合わせて，完全な，あるいはより高度な情報を取得できることが求められる．

これを実現しているのが，複数知識ベースへの横断問合せ [8], [9] を行う手法である．一般的な知識ベース横断問合せにおいては，まず，各知識ベースが持つメタデータに従って入力 SPARQL クエリに含まれる BGP をサブグラフの集合であるサブクエリに分解する．次に各サブクエリを SPARQL クエリへと変換し，それぞれの知識ベースで問合せ処理を行う．最後に，各単一グラフパターンの異種の知識ベースへの問合せ結果を統合して出力する．SPARQL 横断問合せは，各知識ベースの語彙とスキーマを含むメタデータカタログを保持し，サブクエリの各知識ベースへの割り当てをサポートする．

しかし，これらの既存研究ではユーザが SPARQL 構文や知識ベースのスキーマに精通していることを前提としている．提案手法では既存手法と異なり，構文知識を必要としないキーワードクエリから複数知識ベースへの横断問合せを実現する手法を提案する．

3.2 知識ベースに対するキーワード検索

RDF リソースへの問合せは SPARQL によって実現されるが，基礎となる RDF データのスキーマや SPARQL の構文に精通していないユーザにとっては，SPARQL 問合せの作成は困難である．さらに，知識ベースには膨大な量の背景知識が存在するため，熟練者でないユーザだけでなく，専門家にとっても SPARQL クエリを作成することは困難となる場合がある．

そこで，RDF データセットを直感的に検索できるキーワードベースの検索が提案されている．ユーザはキーワードと呼ばれるいくつかの単語を指定することで，システムはその単語に関する知識ベースのリソースの検索を行う．

この達成のために 2 つのアプローチが提案されている．1 つ目は，キーワードから SPARQL クエリの作成を行うことでユーザの要求を満たす問合せを行う方法，2 つ目は，入力されたキーワードを含み、かつユーザの要求を満たす部分グラフを見つけることで問合せを行う方法である．

3.2.1 SPARQL クエリに翻訳する手法

キーワード検索を行う Shekarpour らの手法 [10] や Gkirtzou らの手法 [11] は，ユーザの情報ニーズを捉えた SPARQL クエリ候補集合を自動的に生成し，適切なものをユーザに選択させることによってキーワードからの SPARQL 問合せを実現している．さらに，Wen らの手法 [12] では，キーワード間の文脈情報やキーワードに関連するクラスの情報などをインデックス化しておくことで，より効率的で効果的な問合せを実現している．しかし，いずれの研究でも知識ベースが単一である場合に限定され，専門知識のないユーザが複数の知識ベースの中から問合せを行うデータセットを選択する必要があった．

これに対して，Izquierdo らの手法 [13] では，各知識ベースに対するサブクエリを組み合わせることによって，キーワードから複数知識ベースを横断する SPARQL クエリを構築して問合せをすることが実現されている．しかしながら，利用する全 SPARQL エンドポイントがキーワードを SPARQL クエリに変換することをサポートしている必要があり，これは実際の

4：http://dbpedia.org/
5：https://yago-knowledge.org//
6：http://wikidata.org/

federation 環境では実用的ではない．

また，Wang らの手法 [14] では，事前に各知識ベースが持つ異なるスキーマのクラス間の関係を捉えたスキーマグラフ (スキーママッピング) を作成することで，複数知識ベースを横断するクエリの構築を行う手法を提案している．しかしながら，スキーマグラフをどのように構築するかは大きな課題となる．Wang らの手法 [14] では，異なる SPARQL エンドポイントのスキーマのマージを人手で行うことでスキーマグラフを作成しているが，この処理には非常にコストがかかり，より多くの知識ベースに対して SPARQL 横断問合せを行う場面では実用的ではない．

これに対して提案手法では，事前のスキーママッピングを必要とせず，オンラインで各知識ベースのリソース間の関係を探索することによって，キーワードに関する最小サブグラフの取得を行う．

3.2.2 キーワードに関するサブグラフを抽出する手法

Rihany らの手法 [15], [16] は，キーワードとマッチングした知識ベース内のリソースを集約する問題をシュタイナー木問題 [17] として定式化し，クラスカルのアルゴリズムを適応することで，キーワードクエリに関連するサブグラフの獲得を行う．この手法では，キーワードから SPARQL クエリへの変換を行わないため，事前のスキーママッピングやユーザの介入なしにサブグラフの探索が可能となる．

これらの手法の処理の流れとしては，まず，クエリに含まれるキーワードについて各知識ベースの持つリソースに全文検索を行い，結果として返されたリソースをマッチング要素とする．次に，各キーワードに対するマッチング要素を含む最小のサブグラフを抽出する．最後に，取得されたサブグラフについて，その構成要素からスコアリングを行い，ランキング付けをして結果を出力する．

特に, [15] では，グラフ集約のために DNH(Distance Network Heuristic) [18] を適用し，さらに最小全域木構築でエッジを選ぶ際にグラフの次数中心性を用いることで，より一般的なリレーションを含むサブグラフの構築を達成している．

しかし，既存の研究では問合せ対象が単一の知識ベースに制限されていた．そこで提案手法では，複数知識ベースへのキーワードマッチングと，マッチングされたリソースを含む異種知識ベース間の最小サブグラフの抽出を行うことで，横断的なキーワード検索を行う手法を提案する．

4 提 案 手 法

4.1 提案手法の方針

本研究では，キーワードクエリから複数の知識ベースが持つリソースへの横断的な問合せを実現することを目的とする．手法の方針として，複数知識ベースのリソース間の接続を行うために知識ベース間を横断するリレーションを考慮したサブグラフの探索を行う．具体的には，探索の際に同一エンティティを示す sameAs エッジなどの知識ベースを横断するリレーションを考慮して探索することで各知識ベースに含まれるサブグラフの結合を実現する．

ここで，複数知識ベースを対象とすることによってキーワードとマッチングする要素が増加するため，グラフ集約のコストが増大することが新たな課題となる．そこで，1) 結果サブグラフ内の要素間の距離の最大値として閾値 L を定義して，2 リソース間の最短距離を求めるために必要となる各リソースの到達可能パスの探索を $\lceil L/2 \rceil$ に制限することと，2) リソース間の最短経路を事前に導出することによってサブグラフの問合せ効率の改善を行う．

4.2 処理の流れ

提案システムでは，キーワードクエリを入力として，それに対する解となるサブグラフのランキングを出力として返す．提案手法は 3 つのパートから成り立つ．

まず，第一のステップとして，クエリからキーワードマッチングを行う．次に第二のステップとして，マッチングされた要素を集約してサブグラフを抽出する．ここで，グラフ要素の集約に必要となる完全距離グラフの取得のために，事前にマッチング要素についてのハイパーグラフを構築することでサブグラフ抽出の効率化を行う．最後に第三のステップとして，取得されたサブグラフについてスコアリングを行い，スコアに従ったサブグラフのランキングを結果として出力する．次の項から各ステップについて説明する．

4.3 キーワードマッチング

キーワードマッチングでは，クエリのキーワードについて，各知識ベースの持つリソースに全文検索を行い，結果として返されたリソースをマッチング要素とする．ここで，クエリに含まれるキーワード集合を $K = \{k_1, k_2, ..., k_n\}$ と定義する．また，複数知識ベースの federation 環境を $F = \{G_1, G_2, ..., G_m\}$ とする．各知識ベースは V_j をエンティティ集合，E_j をリレーション集合としたとき，$G_j = (V_j, E_j)$ と表現される．

このとき，クエリに含まれる各キーワード $k_i \in K$ に対して、RDF データグラフ上でそのマッチング要素を全て取得し，得られたマッチング要素集合を $S_i = \{s_{i1}, s_{i2}, ..., s_{in}\}$ とする．マッチングに用いる全文検索では，知識ベースの持つリソースのラベルを対象とし，キーワードと一致する，もしくはキーワードを含む要素を全てを抽出して候補リソースとする．ここで，マッチングは知識ベースに含まれるエンティティとリレーションの双方への紐付けが可能となるが，本研究では簡単のためにエンティティのみを対象とする．

例として，$K = \{Gerry\,Scott, Ridley\,Scott, City\,of\,Wesminster\}$ がキーワードクエリとして入力された場合を考える．ユーザがこのキーワードクエリを発行すると，各キーワードにマッチした要素の集合が図 1 の通り取得できる．

4.4 グラフ要素の集約

キーワードマッチングによって取得された候補エンティティの全組合せについて，各キーワードに対するマッチング要素を含む最小の連結サブグラフを抽出する．得られたサブグラフは入力クエリに対する解となる．マッチング要素の全組合せを考

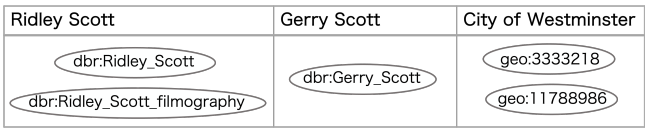

図 1　キーワードマッチングで得られた候補一覧

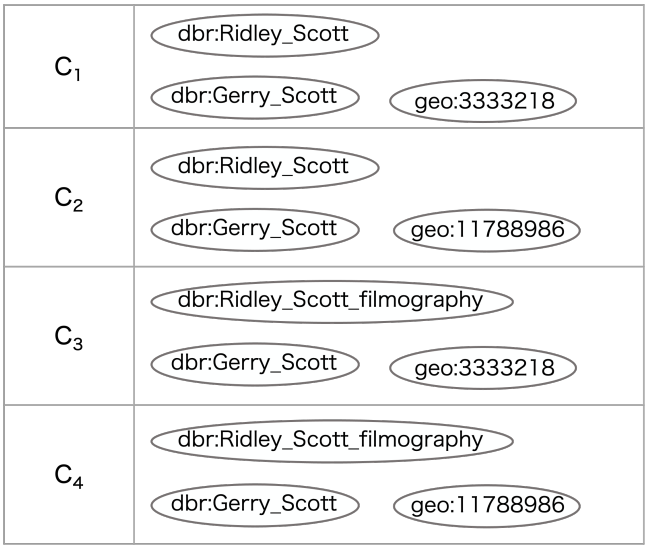

図 2　マッチング要素の全組合せ

えるため，直積をとることで全組合せ $C = S_1 \times S_2 \times ... \times S_n$ を得る．例として，図 1 で得られたマッチング要素の全組合せの例を図 2 に示す．

得られた全組合せについて，各要素を含む最小の連結サブグラフを知識ベースから獲得する．ここで，キーワードとマッチングした知識ベース内のリソースを集約する問題をシュタイナー木問題 [17] として定式化する．知識ベース $G = (V, E)$ について，エンティティの部分集合 $T \subseteq V$ をターミナルとして，エッジの重み付け関数を $d : E \to \mathbf{R}$ としたとき，ゴールは全てのターミナルを含み，重みが最小となる G 内の部分グラフ S を見つけることである．

シュタイナー木問題は NP 困難であるため，DNH(Distance Network Heuristic) [18] を適用する．DNH では，マッチング要素の全組合せについて完全距離グラフを計算し，この最小全域木を求めることで解となるシュタイナー木を獲得する．ここで，既存手法の DNH ではマッチング要素の組合せごとに完全距離グラフを構築するために，各候補リソース間の最短経路の導出を複数回重複して行っていた．さらに，最短経路導出のために，知識ベース全体に対してい各リソースから制限なしの経路探索を行っていた．しかしながら，問合せを行う知識ベースを複数に許容することによりマッチング要素が増加し，グラフ構築のコスト増大が課題となる．そこで提案手法では，最短経路探索の効率化とハイパーグラフ構築による距離グラフ構築効率化の 2 点の改善を行う．

4.4.1　最短経路の探索

既存手法では与えられた 2 つのリソース間の最短経路の導出する際に，探索領域の制限を行わず，知識ベースの全範囲を対象に探索を行っていた．しかしながら，知識ベース内で距離がある程度以上離れているリソース同士は結果として妥当ではなく，解から除かれる必要がある．さらに，複数の知識ベースに対する問合せを許容する場合にはマッチング要素が増えるため，探索量が増大することが問題となる．そのため探索の冗長性を改善する必要がある．そこで，2 リソース間の接続する最大の距離として閾値 L を定め，この距離以内に接続する経路のみを探索することとする．

さらに，SPARQL を用いた最短経路の探索には大きな時間的コストがかかるとされている [19]．これを改善するために，各リソースから距離 $\lceil L/2 \rceil$ 以内に到達可能な全パスの探索を行い，その末端ノード同士の結合を行うことで，効率的な最短経路探索を実現する．

最短経路を求める 2 つのマッチング要素が同一知識ベース内でが見つかった場合にはこの手法で全ての経路の結合が可能となる．一方で，異種知識ベースにおいて見つかった場合には，サブグラフの要素が片方の知識ベースに偏っているため，双方の中央では結合が不可能となる場合がある．

そこで，最短経路探索における結合を行うための場合分けを行う．まず，2 つの頂点が同じ知識ベースに存在する場合には，末端エンティティ同士で全ての経路の結合が可能となる．次に 2 つの頂点が異なる知識ベースに存在する場合を考える．このとき，1) 知識ベースを横断するポイントが中央にある場合には，2 つの頂点が同一知識ベースに存在するときと同様に，全経路について末端エンティティ同士での結合が可能となる．一方で，2) 横断するポイントが中央にない場合には，まず双方の距離 $\lceil L/2 \rceil$ の範囲内に横断ポイントになり得るリソースが存在するかを探し，存在する場合には，($\lceil L/2 \rceil$ – 横断するまでの距離) で結合可能かの判定を行う．横断ポイントになり得るリソースとしては，片方の知識ベースから到達したリソースについて，その URI のドメインがもう一方の知識ベースのドメインと一致するものと定義する．

これによって，全ての 2 リソースを接続するパスを探索し，距離コストが最小となる経路を 2 つのリソース間の最短経路として導出する．ここで，経路長が同一の複数の経路が見つかった場合，問合せに対して最も関連性の高い経路の選択を行う必要がある．そこで，ターミナル間経路の次数中心性の平均を用いることで最適なパスを選択する．経路の次数中心性は次の定義に従って計算される．v_1 から v_n を結ぶ経路を $p = \{(v_1, e_1, v_2), (v_2, e_2, v_3), ..., (v_{n-1}, e_{n-1}, v_n)\}$ とする．このとき，p の次数中心性の重みは $CDW(p) = A(p)$ となる．ここで，$A(p)$ は経路に含まれるノードの次数平均であり，$A(p) = \sum_{i=1}^{n} \frac{deg(v_i)}{n}$ となる．

また，ある要素から到達可能なリソースを探索する場合には，トリプルの持つリレーションとして順方向のパスと逆方向のパスの双方を考えることとする．Algorithm1 に，最短経路探索のアルゴリズムを示す．

Algorithm 1 知識ベースを横断する最短経路探索

Input: 要素 a への到達可能パス集合 $paths(a)$, 要素 b への到達可能パス集合 $paths(b)$, 閾値 L

Output: 要素 a と要素 b の最短経路 P

1: $P \leftarrow []$
2: $minLength \leftarrow L + 1$
3: $maxDegreeCentrality \leftarrow 0$
4: **for** $pa \in paths(a)$ **do**
5: **for** $pb \in paths(b)$ **do**
6: **if** the terminal nodes of pa and pb are equal and the length of the path consisting of pa and pb is less than or equal to $minLength$ and the degree centrality is greater than the $maxDegreeCentrality$ **then**
7: $P \leftarrow Join(pa, pb)$
8: $minLength \leftarrow len(P)$
9: $maxDegreeCentrality \leftarrow GetDegreeCentrality(P)$
10: **end if**
11: **end for**
12: **end for**
13: **if** P is empty **then**
14: **for** $pa \in paths(a)$ **do**
15: **if** the terminal node of pa has the same domain with entity b **then**
16: **for** $pb \in paths(b)$ **do**
17: **if** the terminal node of pb can be reached by distance $(\lceil L/2 \rceil - len(pa))$ from the terminal node of pa **then**
18: $P \leftarrow Join(pa, pb)$
19: $minLength \leftarrow len(P)$
20: $maxDegreeCentrality \leftarrow GetDegreeCentrality(P)$
21: **end if**
22: **end for**
23: **end if**
24: **end for**
25: **for** $pb \in paths(b)$ **do**
26: **if** the terminal node of pb has the same domain with entity a **then**
27: **for** $pa \in paths(a)$ **do**
28: **if** the terminal node of pa can be reached by distance $(\lceil L/2 \rceil - len(pb))$ from the terminal node of pb **then**
29: $P \leftarrow Join(pa, pb)$
30: $minLength \leftarrow len(P)$
31: $maxDegreeCentrality \leftarrow GetDegreeCentrality(P)$
32: **end if**
33: **end for**
34: **end if**
35: **end for**
36: **end if**
37: **return** P

知識ベースを横断する最短経路探索では，要素 a への到達可能パス集合 $paths(a)$ と要素 b への到達可能パス集合 $paths(b)$, サブグラフ内の距離の閾値 L が与えられたときに，要素 a と要素 b の最短経路を出力する．

まず，最短経路 P と最短経路の経路長を持つ変数である $minLength$，次数中心性を持つ変数である $maxDegreeCentrality$ の初期化を行う (1-3 行目)．次に，要素の到達可能パス集合の全組合せについて，パスの末端ノードが同一であり，2 つのパスからなる経路の長さが現在の最短経路の経路長 $minLength$ 以下であり，かつ，次数中心性が現在の最短経路の次数中心性 $maxDegreeCentrality$ よりも大きくなる場合には，最短経路の更新を行う (4-12 行目)．ここまでで，上記の場合分けのうち，2 つの頂点が同じ知識ベースにある場合と 2 つの頂点が異なる知識ベースにあるが横断するポイントが中央にある場合には結合が実行される．

次に，2 つの頂点が異なる知識ベースにあり横断するポイントが中央にない場合の結合を行う．最短経路が空である場合で，かつ，要素 a の到達可能パスの終端ノードと要素 b の持つ URI のドメインが同一である場合には，知識ベースの横断が可能である．このとき，a の末端ノードから b の到達可能パスの各末端ノードまで，a が持つ余剰なパス長，つまり，$(\lceil L/2 \rceil - len(pa))$ で到達できるパスの存在を探索する (13-17 行目)．該当するパスが存在する場合には，最短経路の更新を行う (18-24 行目)．同様の処理を，要素 b の全到達可能パスに対しても行う (25-36 行目)．得られた最短経路 P を結果として出力する (37 行目)．

4.4.2 ハイパーグラフの構築

既存手法では，完全距離グラフ導出のために複数回同じリソースペア間の最短経路の探索を行っている．この冗長性を改善するために，完全距離グラフを構築する前の処理として全マッチング要素間の最短経路を一度求めておき，これを元に距離グラフを構築を行う．これを実現するために，全マッチング要素をノード，マッチング要素間の最短距離をエッジとしたハイパーグラフを構築する．

例示の図 1 のマッチング要素が取得された場合のハイパーグラフの例を図 3 に示す．例示のハイパーグラフでは，3 つのクエリキーワードそれぞれとマッチングした，3 つのマッチング要素集合が存在する．この異なる集合間の要素同士の最短距離と経路の平均次数中心性を重みとして持つハイパーエッジを持つ．

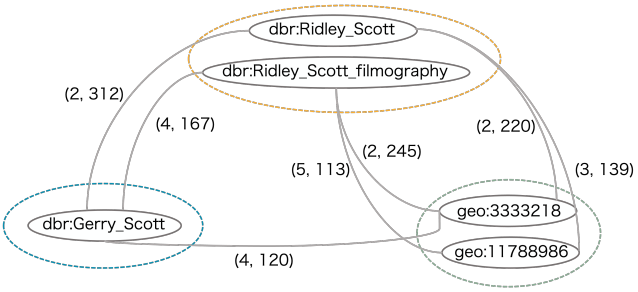

図 3 ハイパーグラフの例

ハイパーグラフの構築のために，まず全要素から閾値 $\lceil L/2 \rceil$ 以内に到達可能な全てのパスの探索を行い，全要素ペアの結合を行うことでエッジを取得する．

4.4.3 シュタイナー木の導出

ハイパーグラフを用いることで，得られたマッチング要素の全組合せについて，各要素を含む最小の連結サブグラフを知識ベースから獲得する．キーワードとマッチングした知識ベース内のリソースを集約する問題をシュタイナー木問題として定式化して，DNH(Distance Network Heuristic) を適用する．DNH では，マッチング要素の全組合せについて完全距離グラフを計算し，この最小全域木を求めることで解となるシュタイナー木を獲得する．以下に，DNH のステップについて説明する．

まず，ハイパーグラフに基づいて，マッチング要素集合であるターミナル T についての完全距離グラフ $DG = (T, E, d)$ を導出する．次に，修正を加えたクラスカルのアルゴリズム [20] を適用することで，完全距離グラフ DG の最小全域木 MST を計算する．クラスカルのアルゴリズムは，グラフの各ノードが個別の木となる森 F(木の集合) を作成し，グラフの全てのエッジを含む集合を構築することからなる．このアルゴリズムでは，森に含まれる任意の木のペアを結ぶエッジの重みである経路長が最小となるエッジから選択して最小全域木に追加する．ここで，距離コストが同一となる複数のエッジが存在する場合には，各経路の次数中心性を考慮して，より適切なエッジ選択を行う．最小全域木が持つエッジの数が $|V|-1$ となるまで，このステップを繰り返す．続いて，最小全域木 MST 内の各エッジを知識ベース内でコストが最小となる経路で置き換えることで，知識ベース G のサブグラフ G' を取得する．最後に，サブグラフ G' に含まれる，次数が 1 となるターミナル T に含まれない末端ノードを削除し，解となるシュタイナー木を獲得する．

例として，図 1 における組合せ C_1 を考える．まず，ハイパーグラフからマッチング要素についての完全距離グラフを図 4 の通り獲得する，完全距離グラフのエッジの重みは，(経路長，平均次数中心性) を示す．次に，クラスカルのアルゴリズムを適用して，最小全域木を構築する．最小全域木構築のステップを図 5 に示す．まず，経路長が最短となるエッジは dbr:Ridley_Scott と dbr:Gerry_Scott を結ぶエッジと，dbr:Ridley_Scott と geo:3333218 を結ぶエッジの 2 本が存在する．そこで，この 2 つのうち，経路の平均次数中心性がより高い dbr:Ridley_Scott と dbr:Gerry_Scott を結ぶエッジを選択し，MST に追加する．続いて，次に経路長が最短で経路の平均次数中心性が最大となる dbr:Ridley_Scott と geo:3333218 を結ぶエッジを追加する．ここで，エッジの本数が $|V|-1$ となるので，これを完全距離グラフの最小全域木とする．MST のノードとエッジを知識ベース G 内の要素と置き換えることによって，解となるサブグラフが得られる．

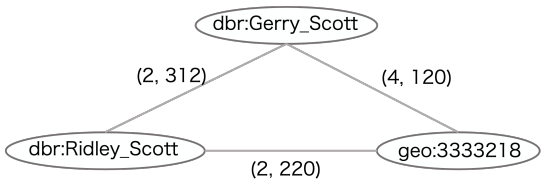

図 4　完全距離グラフの例

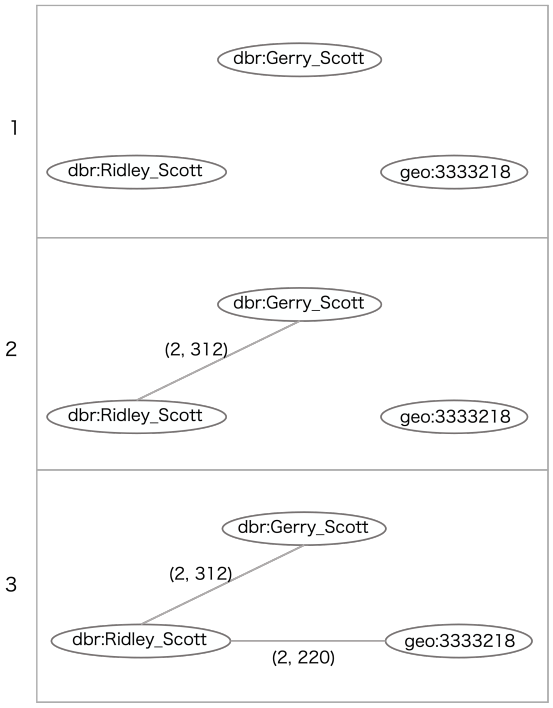

図 5　最小全域木構築のステップ

4.5 結果のランキング

ここまでのステップで，マッチング要素の全組合せについて集約を行ったサブグラフが獲得され，この全てがクエリに対する解となりうる．ここで，これらの全サブグラフについてスコアリングを行うことで，ランク付けを行う必要がある．

既存手法 [15] に従い， 次のように定義されたスコア式に基づいて，結果として得られた全てのサブグラフのスコアリングを行う．

$$Score = 1 - \frac{[w_a * A + (1 - w_a) * L]}{N}$$

ここで，A はマッチングした要素数，L はサブグラフに含まれる A 以外の要素の数，N はサブグラフに含まれる要素の総数，w_a は A についての重みを示す．

スコア式は，直感的に，なるべくキーワードと紐づいたエンティティ以外のリソースを含まないサブグラフをより良いサブグラフとするように定義されている．このスコアに基づいてサブグラフのランキングを行い，ユーザに結果として出力する．

5 評価実験

本実験では，提案手法における複数知識ベースへの展開による問合せ領域の改善と問合せの効率性の向上について評価を行う．

5.1 データセット

構造化知識ベースとしては，DBpedia [4] と GeoNames [5] の 2 つの知識ベースを用いる．DBpedia は Wikipedia を基に知識抽出を行うことで構築された知識ベースであり，GeoNames は全世界の地理データを収めた知識ベースである．どちらも対

象とするデータのドメイン，また，語彙やスキーマの定義が異なるため，保有する情報の種類や粒度が異なる．

本研究では，DBpedia から映画とその監督や出演者に関するリソースを抽出したデータセットと，GeoNames からイギリスに関連するリソースを抽出したデータセットを用いる．各データセットについての統計情報を，表 1 に示す．

表 1　データセットの統計

	DBpedia	GeoNames
Number of entities	278,400	76,979
Number of triples	2,847,843	1,241,034

5.2　実験環境

提案手法の実装には，Python3.8.10 を用いた．実行環境は，Intel® Core™ i7-8700 CPU @ 3.20GHz を搭載した Ubuntu 20.04.3 LTS である．

DBpedia と GeoNames への SPARQL 問合せにはローカル環境に構築した SPARQL エンドポイントとして Virtuoso [7]を用いた．

5.3　評価クエリ

キーワード数が 2〜4 となるクエリをそれぞれ 3 種類ずつ，合計で 9 クエリを作成し，これを用いて評価を行う．

また，このそれぞれのクエリについて，探索の閾値 L を 1〜5 に変化させて問合せを行い，取得された結果の比較を行う．

5.4　比較手法

本研究では，既存のキーワード検索手法の拡張として，複数知識ベースへの問合せを実現することを目的としている．そこで，既存手法を基に，各リソースの到達可能経路探索の際に距離の閾値として定義された L まで探索し，さらに距離グラフ構築の度に最短経路を導出する手法をベースライン手法とする．

一方で提案手法では，最短経路導出の効率化のために各リソースの到達可能経路探索を $\lceil L/2 \rceil$ に制限する．さらに，キーワードマッチング後にハイパーグラフを構築することにより，距離グラフ構築の効率化を行う．

さらに，ベースライン手法に対して最短経路導出の効率化のみを適用した手法を only_threshold，距離グラフ構築の効率化のみを適用した手法を only_hypergraph として，この 2 つの手法についても比較を行う．

5.5　問合せ可能領域

複数の知識ベースを問合せを行うことによる問合せ結果の取得数の変化を観察するために，単一の知識ベースに問合せを行った場合と複数知識ベースに問合せを行った場合での，各クエリについての結果として取得されたサブグラフ数を評価する．結果を表 2 に示す．DBpedia と GeoNames の単一知識ベースを対象として問合せを行った場合をそれぞれ Only DBpedia, Only GeoNames としている．また，DBpedia と GeoNames に対する横断問合せを行った場合を Federation としている．

7：https://virtuoso.openlinksw.com/

結果から，横断問合せを行うことで，単一知識ベースに対する問合せでは取得できなかった，複数知識ベースに跨るサブグラフを獲得できているため，より多くの結果を得られていることが分かる．さらに，ユーザは各クエリに対する知識ベースを選択する必要なく，適切な知識ベースへの問合せが可能となることが示された．

5.6　実行時間

それぞれの手法について L を変化させたときの実行時間の比較を行う．ここで，実行時間が 36,000 秒を超えた場合はタイムアウトとしている．各手法についての，L を変化させた場合の実行時間の比較を図 6 に示す．ここで，ベースライン手法と only_hypergraph 手法についてはキーワード数が 4 語以上の問合せを行なった場合にタイムアウトが発生しているため，ここでは全ての手法について，キーワード数が 2 3 語の問合せの平均実行時間を示す．$L = 5$ の場合，提案手法はベースライン手法の実行時間のおよそ 8 分の 1 程度で問合せを実行できることが示された．

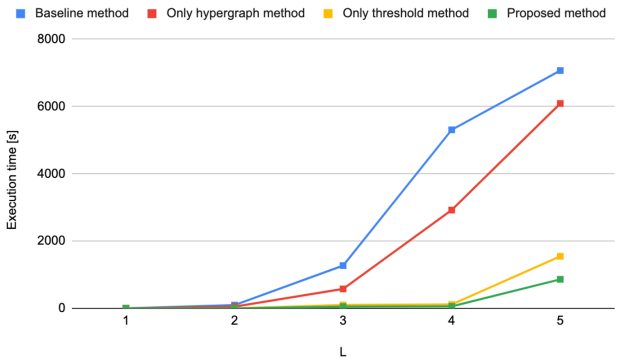

図 6　L を変化させたときの実行時間

5.7　L ごとのサブグラフ取得数

本研究では，既存研究での最短経路探索の冗長性を改善するために，結果サブグラフ内の要素間の距離の最大値として閾値 L を導入した．この L の値を変えたときの結果として得られるサブグラフ数の変化について調査する．結果を表 3 に示す．

本実験で用いたクエリにおいては，$L = 5$ まで探索を行うことで，全てのクエリについて 1 つ以上の問合せ結果を取得することができている．

6　まとめと今後の方針

本研究では，複数知識ベースへのキーワードマッチングと，マッチングされたリソースを含む異種知識ベース間の最小サブグラフの抽出を行うことで，横断的なキーワード検索を行う手法を提案した．実験から，提案手法ではユーザが知識ベースの中から問合せに適切なデータセットを選択する必要なく，キーワード検索が可能となることが示された．さらに，提案手法が問合せ領域のリミテーションを改善し，知識ベースを横断した関係の問合せが可能となることが示された．また，最短経路探

表 2　取得サブグラフ数の比較

	Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8	Q9
Only DBpedia	1	0	0	2	0	0	12	0	0
Only GeoNames	0	0	0	0	0	0	0	0	0
Federation	2	2	1	4	2	3	24	10	3

表 3　閾値 L を変化させたときの取得サブグラフ数の比較

	Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8	Q9
$L=1$	0	0	0	0	0	0	0	0	0
$L=2$	0	0	0	0	0	0	0	0	0
$L=3$	1	0	0	1	0	0	0	0	0
$L=4$	2	2	0	3	2	0	2	2	0
$L=5$	2	2	1	4	2	3	24	10	3

索の効率化とハイパーグラフ構築による完全距離グラフ構築の効率化によって，問合せコストの改善がなされたことについても示された．

今後の課題としては，キーワードマッチングについて，エンティティだけでなく，リレーションやサブグラフ単位への紐付けが可能となるようにマッチングの拡張をすることが考えられる．エンティティと異なり，リレーションは１つの表現に対する同義語パターンを多く存在するため，キーワードとの厳密なマッチングが困難になる．さらに，エンティティと比較して，リレーションは知識ベース内の多くのトリプルで用いられるため，マッチング要素が増大し，これによって結果の候補となるマッチング要素の組合せについても膨大となる．このため，最短経路探索とシュタイナー木導出について，さらなる効率化の検討が考えられる．

謝　　辞

この成果は，国立研究開発法人新エネルギー・産業技術総合開発機構（NEDO）の委託業務（JPNP20006）の結果得られたものです．また，本研究は，(株) 豊田中央研究所（CPI03086）による共同研究経費の助成を受けたものです．

文　　献

[1] Graham Klyne. Resource description framework (rdf): Concepts and abstract syntax. *http://www.w3.org/TR/2004/REC-rdf-concepts-20040210/*, 2004.

[2] Steve Harris, Andy Seaborne, EricPrud'hommeaux. Sparql 1.1 query language. *W3C recommendation*, Vol. 21, No. 10, p. 778, 2013.

[3] André Valdestilhas, Tommaso Soru, and Muhammad Saleem. More complete resultset retrieval from large heterogeneous rdf sources. In *Proceedings of the 10th International Conference on Knowledge Capture*, pp. 223–230, 2019.

[4] Sören Auer, Christian Bizer, Georgi Kobilarov, Jens Lehmann, Richard Cyganiak, and Zachary Ives. Dbpedia: A nucleus for a web of open data. In *The semantic web*, pp. 722–735. Springer, 2007.

[5] Geonames. `https://www.geonames.org/`.

[6] Drugbank. `https://go.drugbank.com/`.

[7] Chebi. `https://www.ebi.ac.uk/chebi/`.

[8] Andreas Schwarte, Peter Haase, Katja Hose, Ralf Schenkel, and Michael Schmidt. Fedx: Optimization techniques for federated query processing on linked data. In *International semantic web conference*, pp. 601–616. Springer, 2011.

[9] Michalis Mountantonakis and Yannis Tzitzikas. Large-scale semantic integration of linked data: A survey. *ACM Comput. Surv.*, Vol. 52, No. 5, September 2019.

[10] Saeedeh Shekarpour. Dc proposal: Automatically transforming keyword queries to sparql on large-scale knowledge bases. In *International Semantic Web Conference*, pp. 357–364. Springer, 2011.

[11] Katerina Gkirtzou, Kostis Karozos, Vasilis Vassalos, and Theodore Dalamagas. Keywords-to-sparql translation for rdf data search and exploration. In *International Conference on Theory and Practice of Digital Libraries*, pp. 111–123. Springer, 2015.

[12] Yanlong Wen, Yudong Jin, and Xiaojie Yuan. Kat: keywords-to-sparql translation over rdf graphs. In *International conference on database systems for advanced applications*, pp. 802–810. Springer, 2018.

[13] Yenier Izquierdo, Marco A Casanova, Grettel García, Frederic Dartayre, and Carlos Henrique Levy. Keyword search over federated rdf datasets. In *ER Forum/Demos*, pp. 86–99, 2017.

[14] Qing Wang, Peng Peng, Tianyao Tong, Zhen Tian, and Zheng Qin. Keyword search over federated rdf systems. In *International Conference on Database Systems for Advanced Applications*, pp. 613–622. Springer, 2020.

[15] Mohamad Rihany, Zoubida Kedad, and Stéphane Lopes. A keyword search approach for semantic web data. In Elisabeth Métais, Farid Meziane, Sunil Vadera, Vijayan Sugumaran, and Mohamad Saraee, editors, *Natural Language Processing and Information Systems*, pp. 131–143, Cham, 2019. Springer International Publishing.

[16] Mohamad Rihany, Zoubida Kedad, and Stéphane Lopes. Keyword search over rdf graphs using wordnet. In *BDCSIntell*, pp. 75–82, 2018.

[17] Frank K Hwang and Dana S Richards. Steiner tree problems. *Networks*, Vol. 22, No. 1, pp. 55–89, 1992.

[18] Lawrence Kou, George Markowsky, and Leonard Berman. A fast algorithm for steiner trees. *Acta informatica*, Vol. 15, No. 2, pp. 141–145, 1981.

[19] Tanvi Chawla, Girdhari Singh, and Emmanuel S Pilli. A shortest path approach to sparql chain query optimisation. In *2017 International Conference on Advances in Computing, Communications and Informatics (ICACCI)*, pp. 1778–1778. IEEE, 2017.

[20] Bernard ME Moret and Henry D Shapiro. An empirical analysis of algorithms for constructing a minimum spanning tree. In *Workshop on Algorithms and Data Structures*, pp. 400–411. Springer, 1991.